Metropolia

Joonas Hälvä 28.5.2012

Tuotantosolun ohjaussovelluksen kehittäminen

Metropolia Ammattikorkeakoulu
Insinööri (AMK)
Tietotekniikan koulutusohjelma
Insinöörityö
28.5.2012

Metropolia

Tiivistelmä

Tekijä(t) Otsikko Sivumäärä Aika	Joonas Hälvä Tuotantosolun ohjaussovelluksen kehittäminen 35 sivua 28.5.2012
Tutkinto	insinööri (AMK)
Koulutusohjelma	tietotekniikka
Suuntautumisvaihtoehto	sulautettu tietotekniikka
Ohjaaja(t)	testaus- ja kalibrointipäällikkö Timo Siirtola lehtori Kimmo Sauren

Tässä työssä on esitelty Vaisalalle uudistuvaan tuotantomalliin kehitetty ohjaussovellus. Uudistuvassa tuotantomallissa tuotteen valmistuksen vaiheet tiivistettiin yhteen soluun, jossa ohjaussovellus toimii. Toiminnallaan sovellus ohjeistaa tuotannon työntekijää sekä on yhteydessä tietokantaan. Ohjelma noutaa tietokannasta tilaustiedot, kerää ja tallentaa tuotteiden jäljitystiedot, sekä antaa pakkaamisen yhteydessä tietokantaan pakkauskomennon.

Työssä on esitelty demo-vaiheeseen saatettu sovellus, koska ohjelman lopulliset vaatimukset eivät olleet sovelluskehityksen aikana vielä lopulliset. Sovellukseen lisättäviä ominaisuuksia ilmaantui jatkuvasti lisää, joten työ rajattiin versioon, jolla onnistuisi tuotannon käynnistys. Jatkokehitys päätettiin toteuttaa myöhemmällä ajankohdalla.

Sovelluksen alustavien vaatimusten joukossa oli ohjelman konfiguroitavuus. Tulevassa tuotantosolussa tuli olla mahdollista valmistaa lähes mitä tuotteita tahansa, joten solun ohjaussovelluksessa kerättävien jäljitystietojen tuli olla säädeltävissä. Solussa on toinen sovellus, joka hoitaa testauksen ja testaustietojen tallennuksen tietokantaan. Vaatimuksena oli myös solun ohjaussovelluksen ja testaussovelluksen välisen tiedonsiirron suunnittelu.

Tuloksena saatiin toteutus, joka toimii hyvänä lähtökohtana jatkokehitysmahdollisuuksien toteuttamiseen. Lean-ajattelun mukaisesti tuotantosolua kehitetään jatkuvasti suuremmilla tai pienemmillä parannuksilla. Näin ollen ohjaussovellusta tullaan tuotantosolun tavoin toteuttamaan jatkuvaa kehitystä, käyttöönoton jälkeenkin.

Avainsanat	C#, .NET, sovelluskehitys, tuotantosolu, ohjaussovellus

Metropolia

Abstract

Author(s) Title Number of Pages Date	Joonas Hälvä Development of control application for production cell 35 pages 28 May 2012
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Embedded Engineering
Instructor(s)	Timo Siirtola, Testing and Calibration Manager Kimmo Sauren, Principal Lecturer

This thesis is about an application that has been developed to control a production line which is being renewed in Vaisala's manufacturing. In the new production line all the manufacturing phases are to be carried out in one production cell. The new cell is where the control application is going to operate. The application instructs the production line worker and is linked to the database. Software acquires the order information from the database, and stores and saves tracking information of the products. When the products are being packed, the software gives the database a packing command.

This paper presents a demonstration version of the software. The requirements for the application were growing so the work was limited to the demonstration version. It was required, though, that the production could be launched without losing any tracking information. Implementing the additional features was left for future consideration.

Among the initial requirements was that the software would be configurable. The new production cell will be used to manufacture different kinds of products so it should be possible to determine what information to collect about products from a configuration file. In addition to the control application, there will be another application operating in the production cell. It was required to plan a solution for applications to exchange information with each other. The final development and testing of the information exchange interface will be performed as further development.

The final demonstration version is a great platform for implementing additional features. Vaisala is renewing their product manufacturing as they are implementing Lean production methods. By definition Lean production is about continuous improvement. Even the smallest improvements could make a great impact on productivity. Therefore, the development of the control application, as well as the production cell, is constant since there is always room for improvement.

Keywords	C#, .NET, software development, production cell, control application

Sisällys

Termit ja lyhenteet

C#	Microsoftin kehittämä oliopohjainen ohjelmointikieli.
CAT2	Uudistuva tietokanta, johon yhdistetään kahden vanhan tietokannan toiminnot.
Lean	Tuotantoajattelumalli, jonka avulla tuotannossa pyritään vähentämään arvoa lisäämättömiä työvaiheita. Arvoa lisääviä vaiheita korostetaan ja pyritään saamaan toiminta mahdollisimman tuottavaksi.
Luotaus	Menetelmä jossa sondi sidotaan palloon, joka nostaa sondin noin 30 kilometrin korkeuteen kunnes pallon tilavuus on kasvanut niin suureksi, että pallo puhkeaa. Luotauksen tarkoituksena on mitata säähavaintotietoja kuten lämpötila, kosteus ja tuuli.
MVC	Sovellusarkkitehtuurityyli, joka muodostuu sanoista model (malli), view (näkymä) ja controller (käsittelijä). Toteutuksen ideana on sovelluksen koodissa erottaa käyttöliittymä muusta sovelluksesta.
.NET	Alusta, joka tarjoaa palveluita sovellusten luomiseen, käyttämiseen sekä suorittamiseen. Erilaisia sovelluksia voi luoda tietokoneelle, verkkoon, erilaiset verkkopalvelut sekä mobiilisovellukset.
Radiosondi	Kertakäyttöinen elektroninen laite, jota käytetään luotauksessa.
Trace2	Vanha tietokantarajapinta valmistettavien tuotteiden jäljitystiedoille.
Vaisala.VTX	Vaisalan luokkakirjasto, joka tarjoaa toteutuksia yrityksen sisäisiin sovelluksiin.
VTX	Vanha tietokantarajapinta testaus- ja kalibrointitiedoille.
VTX2	Uudistuva tietokantarajapinta tuotteiden jäljitystiedoille sekä testaus- ja kalibrointitiedoille.

1 Johdanto

Työ on tehty Vaisalalle osana projektia, jossa on uusittu sondituotannon käytäntöjä ja toimintaperiaatteita. Uudella tuotantomallilla on pyritty selkeyttämään radiosondien tuotantoa sijoittamalla kaikki valmistukseen liittyvät työvaiheet yhteen tuotantosoluun. Tavoitteena on tuotannon yksinkertaistaminen ja asiakkaan näkökulmasta arvoa lisäävän työn korostaminen. Ideana on mukailla Vaisalan noudattamaa Lean-periaatetta. Uudessa solussa on tarkoitus suorittaa kaikki valmistuksen ja testauksen työvaiheet, joita radiosondien tuotannossa on tuotteen kokoamisesta pakkaamiseen.

Insinöörityön aiheena oli kehittää uuteen tuotantosoluun solun ohjaussovellus, joka opastaa käyttäjää tuotannon eri vaiheissa ja tallentaa jäljitystietoja solussa valmistettavista tuotteista. Sovellus vastaa myös tietokantayhteyksistä. Solun ohjaussovellus ottaa vastaan ja tallentaa komponenttien tiedot sekä yhdistää osat lopulliseksi tuotteeksi. Tavoitteena oli luoda sovellus, joka toiminnallaan mahdollistaa tuotteiden sujuvan valmistuksen sekä tallentaa valmistuksen yhteydessä kertyvän jäljitystietojen tietokantaan. Tavoitteena työssä on, että solussa on mahdollista ohjaussovelluksen osalta aloittaa tuotanto. Sovelluksen jatkokehitys tullaan suunnittelemaan ja suorittamaan myöhemmin.

2 Yleiskuvaus

2.1 Työn tausta

Vaisala on havainto- ja mittausalalla palveluita ja tuotteita valmistava, kehittävä ja markkinoiva yritys. Asiakkaat toimivat meteorologian, sääkriittisten toimintojen sekä teollisuuden aloilla. Yrityksen radiosondituotannossa uusitaan tuotteiden valmistusprosessia: kaikki tuotteen valmistuksen työvaiheet tiivistetään samaan tuotantosoluun. Uudessa tuotantosolussa pyritään kehittämään tuotanto mahdollisimman tehokkaaksi, mikä tarkoittaa yksinkertaistetusti enemmän asiakasarvoa mahdollisimman vähällä työllä. Asiakkaan näkökulmasta arvoa lisäävää työtä valmistuksessa on vain tehty kokoonpano- ja kalibrointityö. Tuotannossa on erilaisia vaiheita, jotka eivät sinänsä tuo lisäarvoa lopputuotteelle. Leanissa on tarkoitus karsia näitä arvoa lisäämättömiä vaiheita. Karsittavia ominaisuuksia on esimerkiksi tuotteen edestakaisin siirtelyt, varastointi,

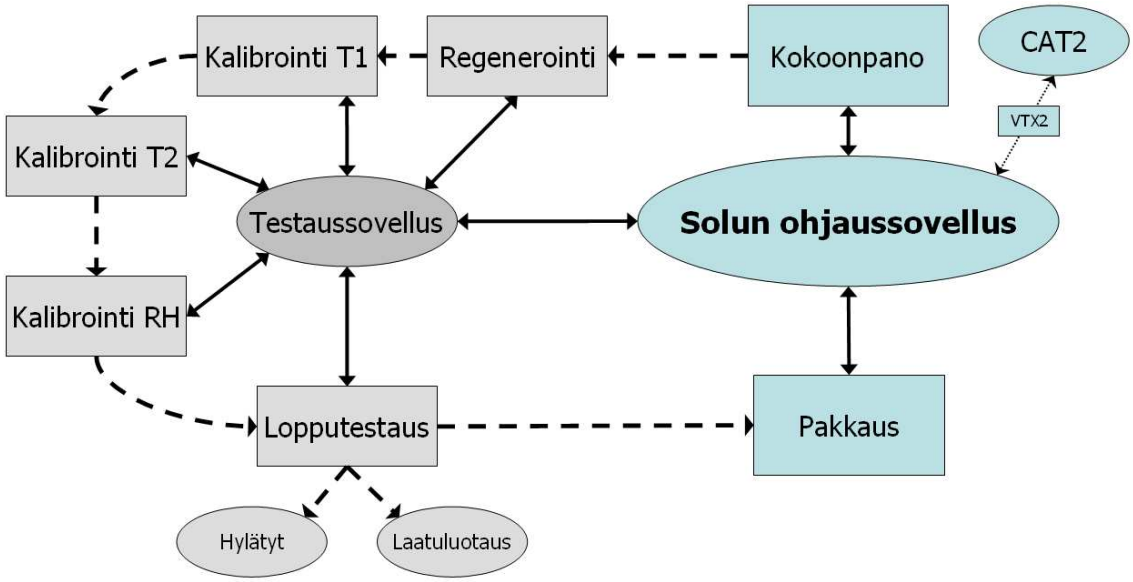
odotusaika ja ylituotanto. Arvoa lisäämättömiä vaiheita pyritään minimoimaan uusimalla tuotannon toimintamalleja tavoitteena lyhentää tuotteen läpivientiaikaa. Se on aika, joka vaaditaan tuotteen jokaiseen valmistusvaiheeseenn osien varastoinnista tuotteen toimitukseen.

Leanin kysyntäperäisessä JIT-mallissa ("just in time") tuotteiden tuotanto käynnistetään vasta todellisten tilausten tai kysynnän perusteella, ja tarjontaperäisessä tuotanto käynnistetään myyntiarvioiden ja tuotantosuunnitelmien perusteella. Kummallakin mallilla toteutetaan Lean-periaatetta ja vältytään helpommin ylituotannolta. Ideana on myös pyrkiä lyhentämään yhden tuotteen läpimenoaikaa, jonka vuoksi solun ohjaussovelluksen vaatimuksissa on, että ohjelma ottaa aikaa ja tulostaa käyttäjälle päivän keskiarvoja ja muita keskeisiä aikoja. Otettujen aikojen avulla voidaan seurata solun tehokkuutta, tuotteiden läpivientiaikoja ja testata solun parannusehdotusten toimivuutta. (Lean Manufacturing Articles and Resources. 2009.)

Radiosondeista kerätään tuotannon yhteydessä vaihtelevasti tuotemallista riippuen jäljitettävää tietoa. Kaiken tämän kerättävän informaation on oltava tallessa tuotteen lähtiessä tuotantolinjalta asiakkaalle. Jäljitystietoihin on tarvittaessa oltava mahdollista palata jälkeenpäin jälkiselvittelyjä varten, mikäli valmistettu tuote palautetaan takaisin viallisena tai muuta selvitettävää ilmenee.

2.2 Työn tavoitteet

Tavoitteena oli luoda radiosondien tuotantoa varten suunniteltavalle tuotantosolulle solun ohjaussovellus, joka toiminnallaan ohjaa käyttäjää valmistettavien sondien määrässä, työnvaiheissa ja tuotetyypeissä. Sovelluksen tuli kuitenkin olla tarpeeksi konfiguroitava, että tuotantosolussa olisi mahdollista valmistaa kaikkia tuotetyyppejä, ei vain jotain tiettyä. Tuotteiden valmistuksen yhteydessä kerättävien tietojen määrä vaihtelee tuotetyypeittäin, joten ohjelman tulee tuotetyypin perusteella määritellä mitä tietoja kerätä. Tuotantosolun toiminta on mallinnettu kuvassa 1.



Kuva 1. Solun toimintakaavio. Katkoviivat kuvaavat valmistettavan tuotteen kulkua solun läpi ja eheät viivat ovat informaation siirtymistä. CAT2-yhteys tapahtuu VTX2-rajapintaa käyttämällä.

Kuva 1 auttaa ymmärtämään tämän työn selostusta tuotannon eri työvaiheista sekä solun toiminnasta. Kuvassa 1 on kaksi sovellusta: testaussovellus sekä ohjaussovellus. Solun ohjaussovellus nimensä mukaisesti on ohjaavassa asemassa solussa tapahtuvissa toiminnoissa ja on ainoa näkyvä sovellus tuotantosolun työntekijälle. Näkymiensä avulla sovellus tulostaa käyttäjälle ohjeistuksia ja näyttää laskurien avulla tuotantotilanteesta.

Ohjelman tuli kerätä talteen kaikki tuotetyyppikohtaisesti tarpeellinen informaatio valmistettavista sondeista, huolehtia tiedon oikeellisuudesta sekä ohjata tuotantoa tarkistamalla, että oikea sondi on oikeassa vaiheessa tuotantosolua. Jokainen valmistettava sondi kiertää u-muotoisen tuotantosolun yksi tuote yhdessä vaiheessa kerrallaan. Tämä reitti on merkitty kuvassa 1 katkoviivoilla. Jokainen solun työvaihe on linkittynyt joko ohjaussovelluksen tai testaussovelluksen kanssa. Nämä yhteydet on merkitty kuvassa eheällä viivalla, jossa nuolet ovat molempiin suuntaan. Sovellus sekä ohjaa kyseistä työvaihetta, että samalla kerää tietoa. Solun ohjaussovellus on myös yhteydessä tietokantaan hakemalla aloitettavan työn tiedot tilausnumeron sekä tilausrivin avulla. Löydettyään tilauksen tiedot päivitetään näytölle kaikki tarvittavat tiedot, jotta tuotanto voidaan käynnistää. Tietokantayhteys on kuvassa 1 merkitty ohuemmalla viivalla, josta käy ilmi käytettävä rajapinta: VTX2.

Solun ohjaussovellus on vastuussa myös tuotannonaikaisesta tallennettavan jäljitystiedon tallennuksesta tietokantaan. Tämä tarkoittaa osien ja lopputuotteen tietojen lisäämistä tietokantaan, eri osien liittämistä tietokannassa toisiinsa ja kaikkien osien linkittäminen lopulliseen lopputuotteeseen. Kaikista seurattavista osista tulee kantaan sisältöä vähintään tuotetyypistä ja sarjanumerosta. Tuotantosolussa ohjaussovelluksen ohessa toimiva testaussovellus suorittaa tuotteille tehtävien kalibrointien ja testauksien yhteydessä kertyvien tietojen tallennuksen tietokantaan. Ohjaussovelluksen ja testaussovelluksen on oltava yhteydessä toisiinsa, jotta molemmat ohjelmat lisäävät varmasti tiedot saman tuotteen tietoihin.

Ajonaikaisilla tarkistuksilla pidetään huolta, että tuotannossa on meneillään niin sanottu one piece flow -tuotanto, jossa useampi sondi kulkee solussa kukin yhdessä työvaiheessa kerrallaan. Kun tuote on koottu, siirretään se uudelleengenerointiin. Generoinnissa ollut tuote siirretään ensimmäisen mittapisteen kalibrointiin ja siinä ollut tuote seuraavan mittapisteen kalibrointiin. Sama siirtäminen jatkuu, kunnes tuote saapuu pakkauspöydälle. Pakattu tuote siirretään pakkauslaatikkoon, ja työntekijä siirtyy ketjun alkuun kokoamaan uutta tuotetta. Lean-mallin mukaisesti otetaan solussa aikaa eri vaiheiden kestoista ja tulostetaan näkymillä tuotteiden läpivientiaikoja ja muita tuotannon seurannassa olennaisia aikoja. Solun ohjaussovelluksen on tarkoitus valvoa tuotannossa olevien sondien määrää ja vaiheistusta, jotta toiminta on edellisiin tuotantomalleihin verrattuna sujuvampaa sekä nopeampaa. Ohjaussovellus pitää samalla huolta, että käyttäjä ei vahingossa sekoita sondien järjestystä.

Radiosondien valmistus on Vaisalalla lähes jatkuvasti käynnissä, joten solun ohjaussovelluksen tulee toimia luotettavasti. Uutta tuotannon toimintamallia käyttöönotettaessa on tärkeää, että sen toimintaa ohjaavan sovelluksen toiminnallisuus on sujuvaa heti alusta lähtien. Käyttöönoton yhteydessä tulisi ilmaantua mahdollisimman vähän ongelmia ja ohjaussovelluksen käytön tulisi olla mahdollisimman ongelmatonta.

2.3 Tuotantosolu

Sondien valmistukseen suunniteltava tuotantosolu on rakennettu siten, että siihen on helppo tuoda laatikoittain tuotteiden osia. Tuotteiden osia tuodaan tarkoituksen mukaisesti tiettyä tilausta varten tarvittava määrä, ja osille on varattu jokaiselle oma lokeronsa tai hyllynsä. Koko solun sijoittelu ja osille tarkoitettujen laatikoiden sijoittaminen on

tarkkaan harkittu suunnittelutapahtumissa, joita kutsutaan kaizeneiksi. Kaizen on osa Lean-ajattelumallia, ja siinä kehitetään asiakassuuntautuneisuutta pienillä jatkuvilla parannuksilla. Kaizen perustuu jatkuvaan kehitykseen, ja jo pienistä hienosäädöistä ja parannuksista voidaan saada suuri hyöty toiminnalle. Kaizenissa tarkastellaan tarkasti koko solun toiminta niin yksityiskohtaisesti kuin mahdollista. Kaikki, mikä voi vähänkin vaikuttaa tuotteen läpivientiaikaan, suunnitellaan tarkasti mahdollisimman toimivaksi. Kaikki solun toiminnat on suunniteltu hyvin tarkasti ruuvinvääntimen sijainnista sekä pöydän optimaalisesta korkeudesta eri laitteenosien laatikoiden asetteluun. (Lean Manufacturing Articles and Resources, 2009.)

Solun suunnittelu on pitkäjänteistä työtä, ja kaikki tuotantosolussa tapahtuva toiminta pyritään suunnittelemaan mahdollisimman sujuvaksi. Uutta solua kehittäessä pyritään luomaan mahdollisimman käyttäjäystävällinen ja toimiva solu, joten tuotannon työntekijöiden panos on solun suunnittelussa tärkeä. Vaikka solussa tulevia työntekijöitä on pyritty kuulemaan solua suunnitellessa, tulee työn ohessa asioita, joita ei tule suunnitteluvaiheessa mieleen. Kaizenin perusajatus on, että työvaiheita pyritään jatkuvasti parantamaan, joten solu tulee kehittymään koko ajan. Kehitysmahdollisuuksia ilmeni sovelluskehityksen aikana, ja niitä ilmenee varmasti lisää myös tuotantosolun käyttöönoton jälkeen.

Solun toiminnan ohjauksesta on vastuussa solun ohjaussovellus, joka ohjaa ja opastaa käyttäjää. Samassa solussa on toiminnassa myös testaussovellus, joka on tuotannossa mukana sondin siirtyessä kokoonpanovaiheesta kalibrointiin. Testaussovellus vastaa kalibroinnissa kertyvän tiedon keräämisestä ja tallentamisesta tietokantaan. Sondin tuotannossa on eri vaiheita. U-muotoisessa solussa suoritetaan kokoonpano, uudelleengenerointi, kalibrointi, lopputestaus ja pakkaus. Kokoonpanossa työntekijä lukee työn alla olevan tilauksen tilaustiedot ja tilausrivin Lean-paperista. Näiden tietojen perusteella haetaan tilaustiedot tietokannasta, ja solun ohjaussovellus tulostaa tilaustiedot näytöille. Samanaikaisesti taustalla on tuotannossa toimiva varastokeräilijä kerännyt tilauksen valmistukseen tarvittavat osat ja toimittanut ne soluun. Kun solun työntekijä on koonnut laitteen ja lukenut viivakoodinlukijalla radiosondin 2D-viivakoodin, on tuote valmis siirtymään solun kalibrointivaiheeseen.

Solun ohjaussovelluksen vaatimuksina on myös, että ohjaussovellus ja testaussovellus vaihtavat informaatiota keskenään. Kun sondi on koottu, lähetetään tuotetiedot

testaussovellukselle ja työntekijä siirtää tuotteen eteenpäin seuraavaan työvaiheeseen. Testaussovellus ottaa tuotetiedot vastaan ja tallentaa kalibrointi- ja testitulokset kantaan. Tuotteen ollessa kalibrointi- ja testausvaiheessa tuotteen piirilevyyn ohjelmoitu ohjelma generoidaan uudelleen, anturit kalibroidaan sekä tuotteen toiminnot käydään kertaalleen läpi lopputestauksessa. Uudelleengeneroinnin yhteydessä testaussovellus lukee tuotteen piirilevyltä tiedot, jotka lisätään lopullisen tuotteen partlist-listaan, eli osalistaan. Osalista on yksinkertaisesti lista tuotteeseen kuuluvien osien tuotetietoja. Piirilevyyn ohjelmoitu sovellus uudelleengeneroidaan, jotta käytössä on varmasti uusin sovellus. Kalibrointivaiheessa tuotteen anturit kalibroidaan kahdessa eri lämpötilapisteessä sekä yhdessä kosteuspisteessä. Tässä vaiheessa tuote voidaan hylätä, jos esimerkiksi kalibrointitulokset eivät ole raja-arvojen sisällä. Lopputestauksessa tarkistetaan tyypillisesti tuotteen langattomien tiedonsiirtojen toimivuus ja käydään läpi kalibroituja arvoja.

Lopputestauksen jälkeen antaa lopputestaussovellus onnistuneesti testatun tuotteen tuotetiedot solun ohjaussovellukselle, joka tulostaa tuotteen pakkaustarrat. Työntekijä asettaa tarrat niille kuuluville paikoille ja pakkaa sondin laatikkoon. Kun pahvilaatikoihin pakattuja sondeja pakataan tilauslaatikkoon, luetaan vielä pakkausvaiheen 2D-lukijalla sondilaatikon viivakoodin. Ohjaussovellus tallentaa kantaan merkinnän pakatusta laitteesta ja päivittää näytölle tilauksen tietoja ja tuotantolaskurien lukemia.

3 Työn vaatimuksia ja toteutus

3.1 Työn rajaus

Tuotantosolua uudistettaessa tuotantosolun asettelun tulee olla selkeä ja ohjelmistojen tulee toimia moitteettomasti. Suunnittelussa mietitään kaikki työvaiheet ja tekemiset hyvin tarkkaan. Koska tuotteiden valmistukseen liittyy niin paljon eri tekijöitä, on myös paljon asioita, joita tulee ottaa huomioon solua ja sen ohjaussovellusta suunnitellessa. Solun ohjaussovelluksen suunnittelu ja vaatimuksien määrittelyt oli aloitettu hyvissä ajoin ennen sovelluskehityksen aloitusta. Vaikka ohjaussovellusta on suunniteltu tarkasti, sovelluksen kehityksen aikana tuli jatkuvasti ilmi sovelluksessa tarvittavia lisäominaisuuksia. Tämä työ päädyttiin rajaamaan ohjaussovelluksen demoversioon, jonka tärkein vaatimus oli, että tuotanto pystyttäisiin aloittamaan uudessa solussa kyseisellä

ohjelmalla. Esiteltynä on ohjelman alustavien vaatimusten ja määrittelyjen esittely sekä vaatimusten mukaisesti toteutettu sovellus. Työstä on rajattu pois Vaisalan sisäinen luokkakirjaston tarkempi esittely, mutta ohjelman oleellisimmat luokat ovat työssä esiteltyinä. Vaisalan VTX-luokkakirjasto esiteltlään yleisellä tasolla, mutta luokkadiagrammit ja toteutukset on rajattu työstä pois.

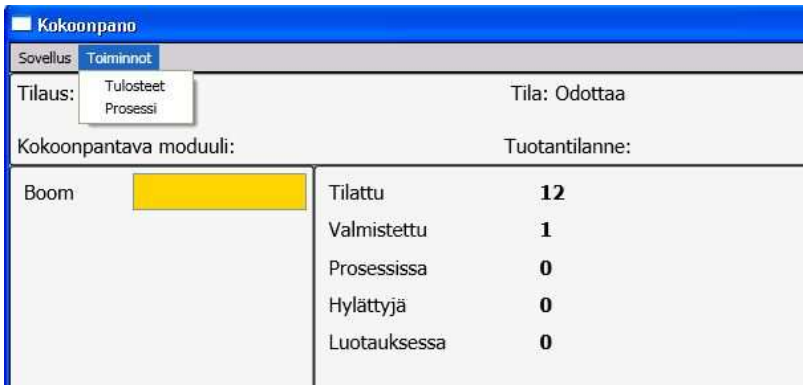
Tässä työssä esitellään demotilaisuuteen valmistettu sovellus, joka on toteutettu alustavien vaatimusten mukaisesti. Tuotteita pystytään valmistamaan solussa menettämättä mitään tietoja. Sovellus on toiminnallisuudeltaan valmis tuotantoon, mutta lisäominaisuuksia on runsaasti. Sovelluksen lisäominaisuuksien lisääminen, testaus, käyttöönotto ja ylläpito tullaan suorittamaan jatkokehityksenä. Jatkokehitysmahdollisuuksia esitellään myöhemmin tässä työssä. Solun ohjaussovelluksen tuli siis olla demovaiheessa projektin kaizenissa eli tapahtumassa, jossa tuotannon vaiheet käydään hyvin tarkasti läpi ja toteutetaan kehitysmahdollisuuksia.

3.2 Solun ohjaussovelluksen vaatimuksia

Tulevassa tuotantosolussa tullaan työskentelemään käytännössä taukoamatta, joten sovelluksen toiminnan vaatimukset ovat tiukat. Vaikka radiosondit ovat kertakäyttötuotteita, tulee niiden toimintavarmuus ja mittaustarkkuus olla maailman kärkitasoa. Solun ohjaussovellus ei toiminnallaan suoranaisesti vaikuta tuotteiden tarkkuuteen tai toimintavarmuuden takaamiseen, mutta toimivalla solun ohjaussovelluksella mahdollistetaan sujuva tuotteiden valmistus. Sovelluksen on toimittava, vaikka valmistettava sondityyppi tai tuotetyyppi olisi eri kuin oli alun perin suunniteltu. Ohjelman tulee siis olla hyvin konfiguroitavissa, jotta ohjelma osaa kerätä tietyn tuotetyypin vaatimia jäljitystietoja ja muita tuotekohtaisesti vaihtelevia tietoja.

solumuotoisessa tuotannossa uutena ominaisuutena tuleva ominaisuus on one piece flow. Kaikki solussa valmistettavat tuotteet käyvät vuorollaan jokaisen työvaiheen läpi yksi tuote työvaiheessa kerrallaan. Solussa etenevät tuotteet eivät saisi ohitella eri vaiheissa toisia tuotteita, ja sovelluksen onkin varauduttava inhimillisiin erehdyksiin. Jos tuotteet menevät sekaisin, on sovelluksen ohjattava työntekijää ja selkeästi esitettävä tuotteiden järjestys. Sovelluksen tulee osata ilmoittaa sekaannuksista ja ohjata sondit takaisin omille paikoilleen tuotantojonossa. Tämä on mahdollista ajonaikaisilla tarkistuksilla ja solun eri työvaiheiden fyysisen sijoittelun selkeydellä. Tuotteiden tietoja

säilytetään ohjelman eri muistipaikoissa, ja tietojen sijainnin perusteella ohjelma osaa tulkita kyseisen tuotteen toivotun sijainnin tuotannossa.



Kuva 2. Kuvassa on ote ohjelman näkymästä tilauksen ollessa työn alla. Kuvassa esitelty kuinka työkalupalkin Toiminnot-valikon alta ilmestyy vaihtoehdot Tulosteet ja Prosessi.

Mahdollisten virhetilanteiden varalta sovelluksen tulostamien tarrojen uudelleentulostaminen on yksi ominaisuus, joka tulee löytyä sovelluksesta. Esimerkiksi jos sovelluksen tulostamaa viivakooditarraa käsiteltäessä tarra rikkoutuu, on oltava mahdollista tulostaa kyseinen tarra uudestaan. Kuvassa 2 on osoitettu, kuinka lisätoiminnot on saatavilla työkalurivin Toiminnot-valikossa olevalla Tulosteet-painikkeella. Kyseisestä painikkeesta avautuu näkymä, jossa on listattuna kaikki ohjelman tuottamat tulosteet, jotka ovat nappia painamalla uudelleentulostettavissa.

Toiminnot-valikosta löytyy ohjaussovellukselle tarpeellinen työkalu työn alla olevien tuotteiden ja tilauksen hallintaan. Sovelluksen vaatimuksina oli, että prosessista täytyy olla mahdollista poistaa tuotteita ja siirtää testaukseen. Vaisala testaa valmistamiaan tuotteita toteuttamalla radioluotauksia testitarkoituksessa, ja nämä tuotteet voidaan ottaa tuotantolinjasta. Sen vuoksi solun ohjaussovelluksessa oli vaatimuksena mahdollisuus vetää tuotannosta tuote luotaukseen.

Tuotteiden poistaminen prosessista onnistuu työkaluvalikon valikosta Prosessi. Objektista aukeaa Prosessinäkymä, jossa on listattuna kaikki työn alla olevan tilauksen yhteydessä alustetut tuotteet. Listasta valitsemalla tai painamalla Lue-painiketta voi valita tuotteen, jolle määrittää tarpeenmukaisia jatkotoimenpiteitä. Tuotteita voi romuttaa tai siirtää laatuluotaukseen testattavaksi. Romuttaminen tarkoittaa yksinkertaisesti tuotteen poistamista prosessista ja luotaukseen laittamisella tarkoitetaan, että tuote otetaan kokoonpanosta sivuun Vaisalan omiin testaustarkoituksiin.

Solun ohjaussovelluksen ominaisuuksiin kuuluu myös kyky vaihtaa informaatiota solussa toimivan testaussovelluksen kanssa. Solun toiminnallisuuden takaamiseksi eri sovellusten välillä tulee olla toteutus rajapintojen väliselle tiedonsiirrolle. Tuotteen tietojen on siirryttävä sovellukselta toiselle, jotta molemmilla sovelluksilla on varmasti oikeat tiedot oikean tuotteen kohdalla. Tuotteen edetessä vaiheesta toiseen siirtyvät myös siihen liitetyt tiedot muistipaikasta toiseen. Kun ohjelmilla on varmasti oikeat tiedot, saadaan kalibroinnin ja testauksen tulokset tallennettua oikean tuotteen tietoihin.

3.3 Solun resurssit

Solussa on tuotannon aikana käynnissä kaksi sovellusta. Toinen on tässä työssä esitelty solun ohjaussovellus ja toinen on testaussovellus. Testauksen suorittama sovellus ei tarvitse näkymiä, joten ohjelmat pyörivät samalla tietokoneella ohjaussovelluksen käyttäessä molempia solun näyttöpäätteitä. Solun ohjaussovelluksen kannalta ainoita syötteitä tuotannon työntekijältä on tietokoneen hiirellä tapahtuvat painikkeen painallukset sekä kahdella viivakoodinlukijalla luettava informaatio. Sekä kokoonpanovaiheella että pakkausvaiheella on molemmilla omat viivakoodinlukijansa. Näillä lukijoilla käyttäjän on tarkoitus lukea tilanteenmukaisia viivakoodeja, joiden sisällön ja ohjelman vaiheen perusteella ohjaussovellus tulkitsee ja toteuttaa toimintoja. Solussa on toiminnassa viivakooditulostin, jonka tulostamat viivakoodit liimataan tuotteiden pakkauslaatikoihin sekä tilauslaatikkoon.

Sekä viivakoodinlukijoilla että tulostimella on kooditasolla omat ajurikirjastonsa. Nämä ovat Vaisalan omaan luokkakirjastoon lisättyjä luokkia, joiden valmiita metodeja kutsumalla saadaan toteutettua toivottu toiminto tai onnistuneesti tulkittua laitteiden vastaanottamaa informaatiota. Vaisalalla on tarkoituksena, että uusien laitteiden kohdalla lisätään kehitetyt ohjelmistoajurit Vaisalan VTX-kirjastoon, josta ne ovat kaikkien käytettävissä. Ajurien avulla saadaan oheislaitteet toimimaan käytettävän laitteiston kanssa, ja yrityksessä jo käytössä olevien laitteiden käyttöönotto on sujuvampaa. Esimerkiksi laitteiden alustukset ja kirjoitus- tai lukutoimenpiteet vaativat vain yhden metodin kutsun ilman syvempää tutustumista laitteen toiminnallisuuteen.

3.4 Työn toteutus

Solun ohjaussovellus on toteutettu C#-kielellä ja .NET-kirjastojen avulla. Vaisalassa on sovelluskehityksessä käytössä myös oma Vaisala.VTX-kirjasto. Tämä on Vaisalan sisäiseen käyttöön tarkoitettu luokkakirjasto, joka tarjoaa valmiita toteutuksia yrityksen tuotanto- ja testaussovelluksissa käytettäviin sovelluksiin. Ohjaussovellus toimii MVC-arkkitehtuurilla (Model eli malli, View eli näkymä ja Controller eli käsittelijä tai kontrolleri), jossa käsittelijä tuntee mallin ja näkymän. Malli ja näkymä eivät tunne toisiaan, eivätkä myöskään käsittelijää. Tämä tarkoittaa sitä, että näkymän ja mallin osat eivät itse vaikuta toisiinsa vaan näkymä päivitetään mallin tietojen mukaisesti käsittelijän antamilla komennoilla. Käsittelijän vastaanottamien käyttäjän syötteiden perusteella ohjataan toimintaa sovelluksen tilanteeseen sopivalla tavalla.

3.5 Resurssit

3.5.1 C#-ohjelmointikieli

Solun ohjaussovelluksen kehitykseen käytettiin C#-ohjelmointikieltä (C sharp). C# on oliopohjainen ohjelmointikieli joka on suunniteltu lisäämään ominaisuuksia ja tukemaan Microsoftin .NET alustaa. C#:ssa ei ole omia luokkakirjastoja, mutta .NET luokkakirjastoista osa on toteutettu C#:n avulla. .NET Framework koostuu kahdesta osasta: CRL:stä (Common Language Runtime) ja luokkakirjastoista (Framework Class Library, FCL), jotka tarjoavat modernien ohjelmointikielten tarvitsemat palvelut. CLR hoitaa kaikki laitteiston ja käyttöjärjestelmän välisen tiedonvälityksen sekä ajonaikaisen ohjelmistokoodin hallinnan. FCL on laaja objektipainotteinen kirjasto, joka tarjoaa rajapinnat, joilla voi toteuttaa CLR:n tarjoamia palveluita. (.NET Framework 4. 2012.)

3.5.2 Vaisala.VTX-luokkakirjasto

Uuden tietokantarajapinnan myötä päivittyy myös Vaisala.VTX. Se on luokkakirjasto, joka sisältää valmiita metodeja yrityksen sisäisiin sovelluksiin. Ydinrajapinta jakautuu kolmeen keskeiseen osioon: VTX.Configurationiin, VTX.Instrumentiin sekä VTX.Dataan. Muita rajapinnan komponentteja ovat käyttöliittymän luontiin liittyviä toteutuksia, testikontrolleri sekä taustasuoritustoteutukset. Background taskeilla eli taustalla ajettavilla tehtävillä onnistuu ohjelman toimintojen suorittaminen erillisesti tausta-ajona. Taustalla

suorittaminen tapahtuu siis ohjelman muusta toteutuksesta rinnakkaisella prosessisäikeellä, eli toteutuksia ajetaan rinnakkain. Hyvä esimerkki taustalla suoritettavasta toteutuksesta ovat ohjaussovelluksessa toteutetut tietokantayhteydet. Tietokantaan annettavien komentojen sekä hakujen toteutus tapahtuu rinnakkaisella säikeellä, koska muuten koko ohjelman suoritus tuntuisi olevan jumiutunut komentojen tai hakujen ajan.

Solussa tapahtuva kahden sovelluksen välinen tiedonsiirto toimii myös taustasuorituksena. Taustalla pyörii kaksi säiettä, joista toinen odottaa, että siihen asetettu lähetettävä tieto kuitataan vastaanotetuksi. Toinen taustasäie odottaa testaussovellukselta lopputestauksesta saapuvan tuotteen tietoja. Toteutus saatiin kehitettyä, mutta jatkokehitys ja testaus jäivät tehtäviksi jatkokehityksen yhteydessä.

Configuration-osio tarjoaa työkalut kehitettävän sovelluksen joustavaan konfiguroitavuuteen. Työkalujen avulla pystyy noutamaan ja yhdistämään asetustiedot useammasta konfiguraatiotiedostosta. Osiota käytetään lataamalla asetukset muistiin, jonka jälkeen ne ovat käytettävissä current-avainsanalla. Tyypillisesti ohjelman alustuksen yhteydessä ladattavat konfiguraatiotiedostot ovat ennalta määritellyn muotoisesti tuotettuja XML-tiedostoja. Tiedon haku tiedostoista tapahtuu avainsanojen avulla. Tiedostot voivat myös viitata toiseen asetustiedostoon, jolloin konfiguraatiotietoja haettaessa etsitään avainsanoja jokaisesta viitatusta asetustiedostosta. Ajonaikaisesti saatua tietoa on mahdollista tallentaa konfiguraariotiedostoihin, jos esimerkiksi jotkin asetukset muuttuvat ajon aikana ja ne tulee saada talteen.

```
<?xml version="1.0" encoding="utf-8" ?>
<config xmlns="http://vaisala.com/vtx/configuration"
  xmlns:c="http://vaisala.com/vtx/configuration/custom">
  <c:include path="configuration2.xml" />
  <section1 key1="value1" key2="value2" />
</config>
```

Kuva 3. Kuvassa esimerkki konfiguraatiotiedostosta.

Kuvassa 3 olevassa esimerkkitiedostossa on kyseisen tiedoston rinnalle hakurakenteeseen liitetty configuration2.XML asetustiedosto. Tiedoston nimiavaruudessa on oltava viittaus VTX/Configuration/Customiin, jotta konfiguraatiotiedostossa voidaan viitata

toiseen tiedostoon. Tiedostojen sisällytystoiminto kuuluu kyseisen nimiavaruuden toimintoihin. Tämä esimerkkitiedosto on hakujärjestyksessä ensimmäisenä, jonka jälkeen haluttuja tietoja haetaan tarvittaessa seuraavasta tiedostosta. Tiedostoista haku tapahtuu ensin rajaamalla haluttu osio, joka tässä esimerkissä olisi "section1". Valitusta osiosta haetaan tietyn avaimen arvoa. Tässä esimerkissä on kaksi avainta, joista hakea: key1 ja key2.

Tietotekniikassa ohjelmistot ovat yhteydessä ulkoisiin laitteisiin ajureiden avulla. Laiteajuri on ohjelmistokoodia, joka on rakennettu ohjaamaan tietokoneen oheislaitteistoa. Laiteajurit tarjoavat käyttöliittymän laitteistolle ja voivat toteutuksellaan erottaa samanlaiset laitteet toisistaan. VTX.Instrument tarjoaa yrityksen sovelluksissa käytettävien laitteiden ajureita, joita sitten voi tarvittaessa käyttää.

VTX.Data:n toimintoja käytetään testattavien sekä valmistettavien tuotteiden tietojen tallennukseen ja hallintaan. Osion toiminnoilla kerätään ajonaikaisesti tuotettua tietoja, kuten sarjanumerot, tuotetiedot ja testitulokset. Osio tarjoaa myös ratkaisut tietojen talletukseen tietokantaan sekä tietokannasta hakemiseen.

VTX-kirjasto tarjoaa metodit kaikelle kannan ja testi- sekä kokoonpanoasemien väliselle liikenteelle. Edeltävissä kantayhteysrajapinnoissa tietojen tallennus on toteutettu isäntäprosessien käynnistämillä prosesseilla. Testiasemakohtaisilla prosesseilla on yksittäinen tiedosto, johon syötetään tallennettavia tietoja ja komentoja. Uudessa VTX2 rajapinnassa jäljitys ja mittadatan tallennus on toteutettu kokoonpano-, testi- ja kalibrointiasemien lähdekoodissa. Tietokantapalvelun isäntäprosessi on toteutettu IIS-palveluna. Tuotantosolusta viitataan ohjelmistopalvelimen palveluun määrittelemällä konfigurointitiedostossa URL-muotoinen palveluosoite. Sovelluksista kutsutaan VTX-kirjaston CatServiceProvicer-luokan metodeja, jotka suorittavat haluttuja toimintoja palvelun kautta tietokantaan. IIS (Internet Information Services) on Microsoftin palvelinohjelmistokokonaisuus. IIS on Microsoftin kehittämä palvelinohjelmistokokonaisuus verkkosivujen ylläpitämiseen Windows-pohjaisilla palvelimilla. VTX2-tietokantayhteyden suoritusohjelmistoon viitataan siis verkkosivun välityksellä. Tämä verkkosivu on toteutettuna tietokannassa IIS-palveluna, jonka URL-osoite on yksinkertaisesti ohjelmistopalvelimen nimi sekä käytettävä portti.

3.5.3 VTX2-tietokantarajapinta

Vaisalalla ollaan uusimassa tietokantarakenne, ja sen myötä uudistuvat tietokantarajapinnat. Jäljitysdatan Trace2-tietokantakanta sekä mittadatan CAT-tietokanta yhdistetään yhdeksi CAT2-tietokannaksi. Aikaisemmin oli kaksi rajapintaa kannan ja testi- sekä kokoonpanoasemien välisiin tiedonsiirtoihin: jäljitystietojen Trace2 sekä mitta- ja kalibrointitietojen CAT. Nämä oli toteutettu rajapintakohtaisilla keskustelutiedostoilla, joihin annetuilla komennoilla käynnistettiin testi- tai tuotantoasemakohtaisia prosesseja. Näiden prosessien toiminta perustui testiasemakohtaisten MES- tai MEA-tiedostoissa tapahtuvien muutosten seuraamiseen sekä tiedoston sisältöjen tulkintaan.

Trace2 rajapinnan testiasemakohtainen MES-tiedosto sisältää toteutettavien toimintojen komennot ja tuotteiden tai tilausten tietoja. Trace2-puolen komentoja on esimerkiksi tuotetietojen lisääminen, pakkauskomento, toimituskomento, tietojen kommentointi sekä erilaiset tietojen hakemiset ja tietojen korjauskomennot. VTX:n puolella käynnistetään testiasemalle oma prosessi joka seuraa testiasemakohtaisessa MEA-tiedostossa tapahtuvia muutoksia. Tässä tiedostossa on runsaasti informaatiota tuotteesta, tuote-erästä, tarkat määritelmät suoritetuista testeistä ja kalibroinneista sekä itse mitta- ja kalibrointitulokset. Nämä edeltävään tietokantaan sisältöä tuottavat rajapinnat uusittiin uuden tietokannan myötä. Toiminnallisuudeltaan rajapintoihin ei tule suurempia muutoksia, mutta toteutukseen tuli hieman korjauksia, jotta rajapinnat osaavat tuottaa sisältöä yhteen kantaan kahden sijasta.

Uudistuksen myötä Vaisala.VTX-luokkakirjastoon lisätään uusi VTX2-tietokantarajapinta. Tämän rajapinnan myötä uudistuu toiminta siten, että kaikki tietokantayhteys toteutetaan suoraan kokoonpano- ja testausasemien sovellusten lähdekoodissa käytettävillä VTX:n tarjoamilla metodeilla. VTX-kirjasto CatServiceProvider-luokan, jossa on uuden tietokantarajapinnan toteutus.

Solun ohjaussovellus tulee olemaan ensimmäisiä tuotantosovelluksia, jossa hyödynnetään uutta VTX2-rajapintaa. Täysin uudenlainen toteutus tuotannossa antaa esimakua uuden rajapinnan tuomista eduista vanhoihin verrattuna.

4 Solun ohjaussovellus

4.1 Sovelluksen esittely

Solun ohjaussovellus on uudistuvassa tuotantosolussa ainoa käyttäjälle näkyvä sovellus. Toiminnallaan sovellus ohjaa toimintaa solussa ja tallentaa keräämiänsä tietoja tietokantaan. Valmistettava tuote kiertää työvaiheesta toiseen, ja käyttäjän antamien syötteiden mukaisesti ohjaussovelluksen tallentamia tietoja siirretään eri muistipaikoista toiseen. Tuotteille on varattu muistipaikkoja eri luokista ja tuotteen siirtyessä vaiheesta toiseen siirtyvät myös tuotteen tiedot asianmukaisesti ohjelman sisäisesti.

Ensimmäisenä ohjelman käynnistyessä ohjelma pyytää käyttäjää kirjautumaan. Tämä on toteutettu yksinkertaisella näkymällä, jossa kysytään käyttäjän tai käyttäjien tunnuksia. Tietoja käytetään pakkauksen ja testausten yhteydessä. Kantaan tallennusten yhteydessä syötetään myös parametrina tieto siitä, kuka on kyseisen toiminnon suorittanut. Kirjautumisen jälkeen ohjelma odottaa uutta tilausta otettavaksi työn alle. Tilaus otetaan valmistettavaksi lukemalla viivakoodinlukijalla tuotantosoluun tuodusta tulosteesta tilausnumero ja tilausrivi. Näiden perusteella solun ohjaussovellus tulostaa näytölle tietokannan palauttamat tilaustiedot ja käynnistää tuotannon. Jos tietokannasta ei löydy annettua tilausta, ilmoitetaan siitä työntekijälle.

Tuotannon ollessa käynnissä odotetaan tuotteeseen kuuluvien seurattavien osien viivakoodien lukemista. Tuotteen tiedot tallennetaan ohjelman muistiin niille tarkoitettuihin paikkoihin. Käyttäjän on tarkoitus lukea tuotteen osien viivakoodit samalla kun hän kokoaa lopullista lopputuotetta. Kun tuotekohtaisesti määritellyt seurantatiedot on otettu ohjelmassa vastaan ja tuote koottu, opastaa ohjaussovellus työntekijää siirtämään tuote kalibrointi- ja lopputestausvaiheeseen. Seuraavassa vaiheessa ollut laite siirretään seuraavaan vaiheeseen, ja sama vaihtoprosessi tapahtuu aivan solun kierroksen loppuun asti tai kunnes solussa on vaihe tyhjillään. Ideana on lopulta täyttää kaikki vaiheet ja saada täysi tuotanto käyntiin. Kalibrointi- ja lopputestausvaiheessa on vaiheita uudelleengenerointi, kalibroinnit kahdessa lämpötilassa sekä kosteusmittapisteessä ja lopputestaus.

Kun tuotteen lopputestaaminen on valmistunut, kutsuu testaussovellus ohjaussovelluksen toteutusta, joka odottaa saapuvan tuotteen tietoja. Kun tuotteen tiedot on

vastaanotettu, ohjataan ohjelman suoritus toteutukseen, joka tulostaa ilmoituksen pakkausvaiheen näkymässä sekä siirtää testauksesta tulleen tuotteen tiedot määriteltyyn muistipaikkaan. Metodissa tulostetaan myös pakkauspöydällä sijaitsevalla tulostimella kyseiselle tuotteelle tarra, josta käy ilmi tuotetyyppi ja sarjanumero. Tuote pakataan sille suunniteltuun pahvilaatikkoon ja sijoitetaan foliopussiin. Foliopussista imetään vakuumi-imurilla ilmat pois, jonka jälkeen tuote on valmis pakattavaksi tilauksen pakkauslaatikkoon. Jos tuote on ollut pakattavaksi tullessaan pakkauslaatikon täyttävä tuote, tuotteen omaa tarraa tulostaessa on tulostettu myös pakkauslaatikon tarra. Kun vakuumi-imuri on valmis, lukee työntekijä pakkauspöydän viivakoodinlukijalla sondilaatikon viivakoodin. Viivakoodinlukijan tapahtumankäsittelijä ohjaa sovelluksen suorituksen metodiin, jossa tuote lisätään pakattavien tuotteiden listaan. Jos pakkauslaatikko täyttyy tuotteesta, kutsutaan tietokantapalvelun luokkakirjaston metodia pack, joka antaa tietokantaan ilmoituksen pakatuista tuotteista. Näkymät päivitetään kuvaamaan tuotannon tilannetta, ja tuotannon työntekijä palaa kierrossaan kokoonpanovaiheeseen.

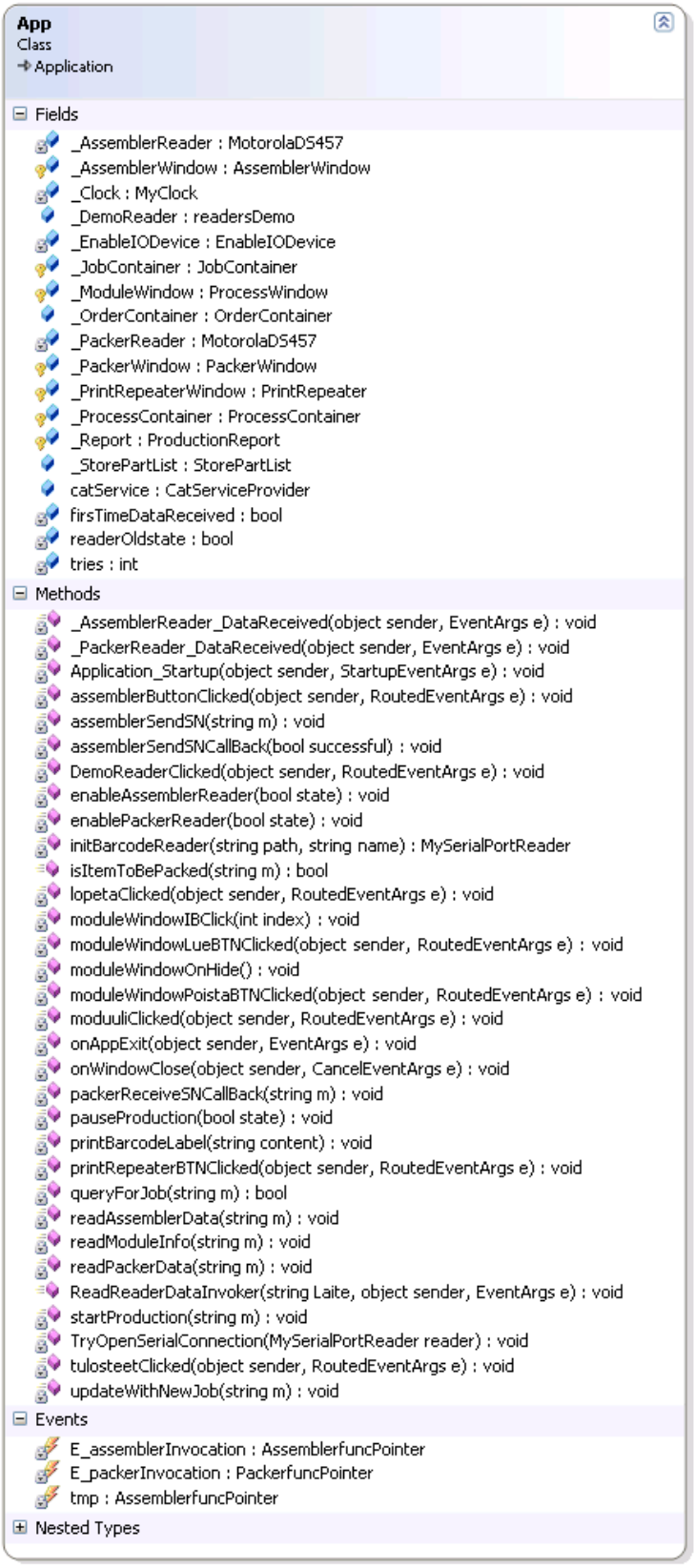
4.2 Alustusfunktio

Ohjelman käynnistysfunktiossa ladataan konfiguraatiotiedostosta tietokantapalvelun URL-osoite. Samankaltaisesta XML-tiedostosta ladataan ohjelman käyttämien instrumenttien, kuten tulostimen ja 2D-lukijoiden, asetuksia. Nämä XML-tiedostojen sisällöt ovat rakenteiltaan ennalta määriteltyjä, jotta VTX-luokkakirjaston sisäiset latausmetodit osaavat ladata tarvittavat konfiguraatiot ohjelman käyttöön. Tässä työssä ladattavia asetuksia ovat laitteiden kohdalla esimerkiksi käytettävän instrumentin nimi, yhteyskohtaisia asetuksia sekä käytössä oleva COM-portti. Yhteyskohtaisia asetuksia on tyypillisesti siirtonopeus, databittien määrä, pariteetti ja lopetusbitin määritys. Kun instrumenttiasetukset on ladattu, alustetaan viivakoodinlukijat ja luodaan ohjelman käyttöliittymä. Viivakoodinlukijoiden sekä näkymään liittyvien tapahtumien tapahtumankäsittelijät luodaan sekä määritellään. Kun kaikki toiminnallisuus on alustettu ja toiminta valmis aloitettavaksi, tarkistetaan vielä, onko edelliseltä istunnolta jäänyt tilausta työn alle.

Edellisen istunnossa mahdollisesti kesken jäänyt prosessi on tallennettuna session-tiedostoihin. Nämä tiedostot ovat AssemblerReader.bin, Clock.bin, JobContainer.bin, PackerReader.bin, StoreUutPartList.bin, OrderContainer.bin ja ProcessContainer.bin. Ohjelmaa alustettaessa määritellään, että ohjelman sulkeutumisen yhteydessä

tarkistetaan sen hetkinen tuotantotilanne ja tallennetaan tarvittaessa tiedot tiedostoihin. Työssä myöhempänä esiteltävien muistien sisällöt tallennetaan tiedostoihin, jos tilaus on vielä työn alla.

Jos edellisen istunnon tietoja ei ole, alustetaan ohjelma valmiiksi vastaanottamaan tilauksen tietoja. Ohjelma pyytää käyttäjää lukemaan kokoonpanopöydän viivakoodinlukijalla tilauksen numerotunnuksen ja rivin. Ohjelman alussa tarkistetaan, että sovellus on yhteydessä tietokantaan, ennen kuin käyttäjä saa luvan aloittaa työnteon. Kantayhteyden tarkistaminen tapahtuu kutsumalla VTX-luokkakirjaston CatService-luokan metodia, jonka palauttama arvo määrittää yhteyden. Tarkistuksen jälkeen asetetaan kokoonpanon 2D-lukija aktiiviseksi, jonka jälkeen ohjelman toiminta on tapahtumankäsittelijöiden varassa. Tapahtumankäsittelijä on ohjelman arkkitehtuurissa toimintaa hallitseva osa, joka muokkaa käyttäjältä saamien käskyjen perusteella näkymää ja mallia.

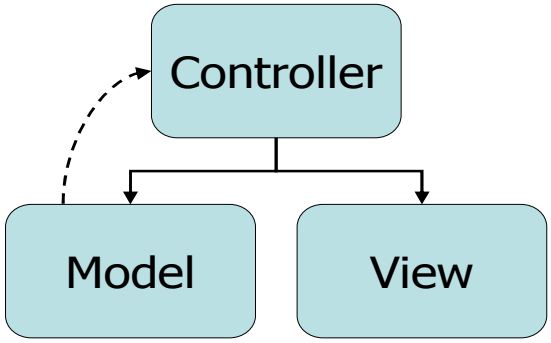


Kuva 4. Luokkadiagrammi ohjelmaluokasta, jossa on metodit ja ominaisuudet joilla ohjelman suoritusta ohjataan.

App-luokassa ovat kaikki metodit, delegaatit ja tapahtumankäsittelijät, joilla saadaan suoritettua sovelluksen toiminnot. Kuvan 4 luokkadiagrammi osoittaa App-luokan sisällön monipuolisuuden. Luokassa on toteutuksia, joiden avulla ohjelma alustetaan ja käyttäjän antamat syötteet saadaan tulkittua halutulla tavalla, sekä delegaatiot, joiden avulla syötteet tulkitaan. Tapahtumankäsittelijä tulkitsee delegaatin pohjalta tarkoituksen mukaisen metodin ja suorittaa sen. Suorituksen yhteydessä tulostetaan näkymille mahdollisia informaatiotulosteista ja päivitetään mahdollisesti muuttuneiden tuotantolaskurien lukemat näkymille.

4.3 Sovelluksen rakenne

Sovelluksen toiminta on jakautunut useampaan luokkaan, joissa kukin toteuttaa erinäisiä toimintoja, kuten varastoi tietoa, muokkaa näkymiä tai tulkitsee käyttäjän antamia komentoja. Sovellus on toteutettu MVC-arkkitehtuurilla (Model-View-Controller), joka muodostuu osista malli, näkymä ja käsittelijä. Ohjaussovelluksen luokat esitellään arkkitehtuurin osien esittelyjen yhteydessä. Ohjelman alussa on alustusfunktio, jossa ladataan konfiguraatiotiedostoja ohjelman muistiin sekä alustetaan laitteita, näkymiä ja tapahtumankäsittelijöitä. Sen jälkeen kontrolleri on ohjelman toiminnasta vastuussa. Kuvassa 5 havainnollistetaan, kuinka MVC-arkkitehtuurin kontrolleri on ohjelmassa ohjaavassa asemassa. Kontrolleri tulkitsee käyttäjän antamia käskyjä ja sen mukaisesti joko päivittää tai noutaa tietoa malliosion muistipaikoista sekä päivittää näytölle ilmoituksia tai muistista saatuja tietoja.



Kuva 5. MVC-arkkitehtuurin idea havainnollistettuna. Kontrolleri hallitsee malli- ja näkymäosioita. Malli palauttaa kontrollerin pyytämiä tietoja.

App-luokka on ohjelman suoritusta hallitseva luokka. Siitä kutsutaan omien metodien lisäksi toisten luokkien metodeihin ja ominaisuuksiin. Toiseen luokkaan viitataan

luomalla olio, jonka avulla onnistuu toteutusten kutsuminen. Olio on luokan esiintymä, eli muuttuja jonka tyyppi on toinen luokka. Ohjelman luokat on luotu juuri ohjaussovellusta varten. Ohjaussovelluksen luokat esitellään työssä tarkemmin MVC-arkkitehtuuriosioiden avulla. Lisäksi sovelluksessa on käytössä Vaisalan oma luokkakirjasto, jota ei tässä työssä esitellä kuin toteutuksista kertomalla. Tarkempi luokkien esittely on rajattu pois.

4.3.1 Malliosio

Model eli malli edustaa ohjelman ajon aikaisten tietojen säilytyspaikkoja. Valmistettavien laitteiden edetessä tuotantosolussa kerätään siitä talteen tärkeätä tietoa ohjelman malliosion muistipaikkoihin. Jäljitysdatan tallennus on tärkeätä, koska voidaan joutua esimerkiksi jäljittämään tietyn tuote-erän tuotteet tai jotain muuta jälkeenpäin selvitettävää voi ilmetä. Kuvassa 6 esitellään eri tyyppejä, joita luokassa voi C#-kielessä olla. Näiden esimerkkien avulla luokkien esittelyjen yhteydessä käytettävät termit ovat selkeämpiä. Sovelluksen rakenteessa on useampia malliosioon kuuluvia luokkia, joissa jokaisessa on muuttujia, joihin tieto sijoitetaan, sekä metodeja, joilla luokkien sisältöä saa muokattua, noudettua tai poistettua. Luokissa on ominaisuuksina ja kenttinä kerättävien tietojen tyyppi ja nimi. Luokissa on myös muodostimia, joiden avulla eri luokissa käytettäviä ominaisuuksia saadaan luotua toisen luokan ominaisuuksien pohjalta.

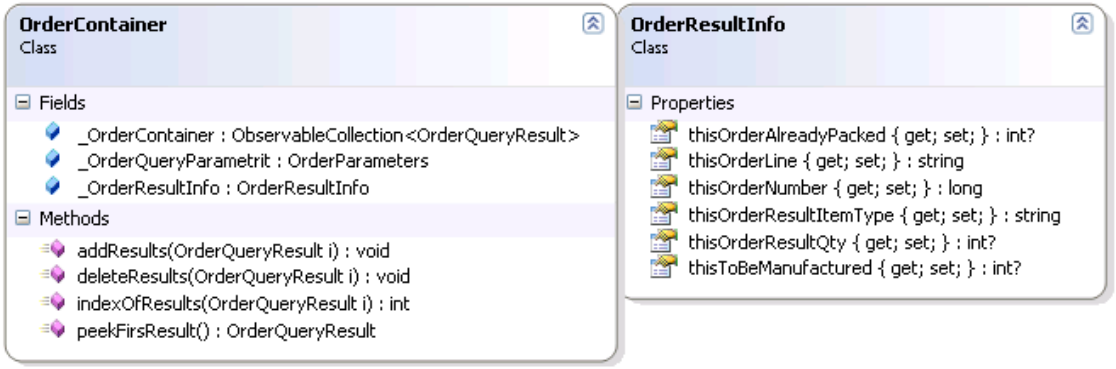
```
class Tuote
    {
        // Ominaisuus
        public string TuoteTyyppi { get; set; }
        // Kenttä
        public DateTime Valmistettu;
        // Muodostin
        public Tuote()
        { }

        // Metodi
        public void AnnaTuote()
        {
            Console.WriteLine("Tuotetyyppi on " + TuoteTyyppi);
        }
    }
```

Kuva 6. Esimerkkiluokka "Tuote", jossa on neljä eri jäsentä: ominaisuus, kenttä, muodostin ja metodi.

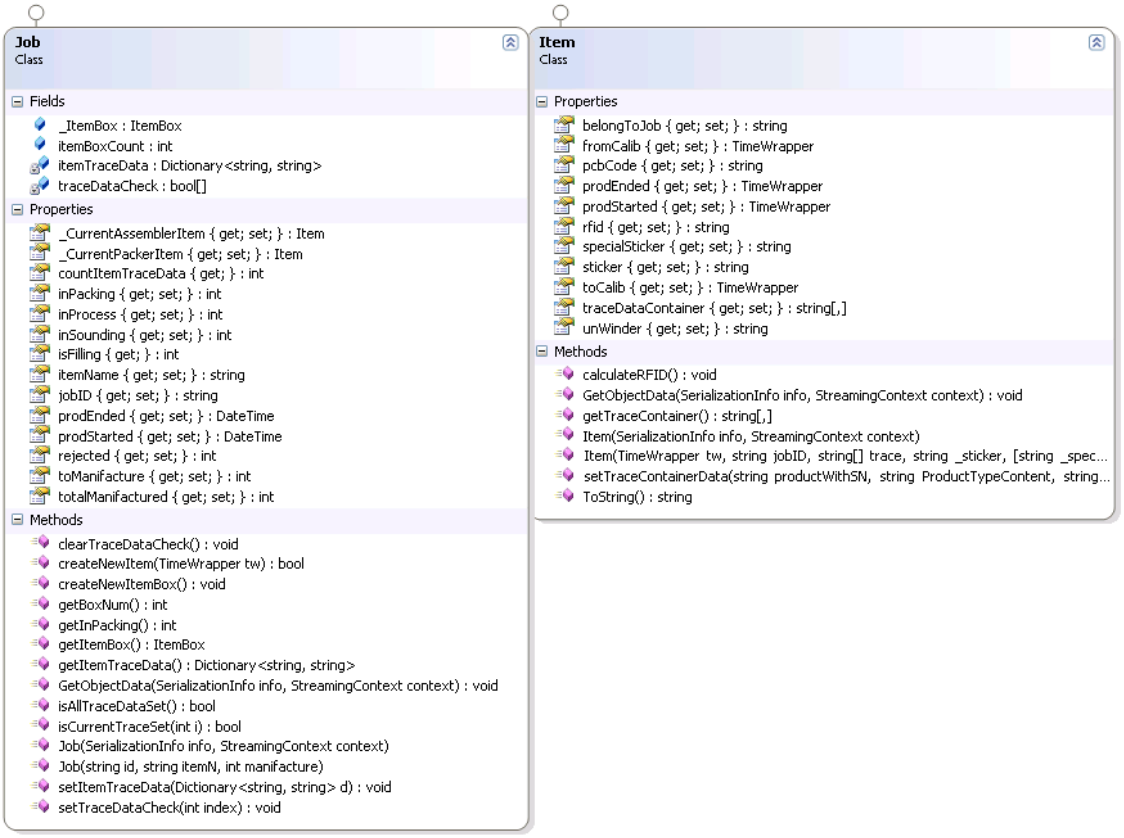
Tässä alaluvussa esitellään malliosion luokkia. Luokat sisältävät siis ominaisuuksia, kenttiä, muodostimia ja metodeja, joiden avulla tallessa olevia tietoja voidaan hakea tai

asettaa App-luokan toimintojen avulla. Tallennettavien tietojen ominaisuuksien rakenteet voivat muodostua toisen luokan ominaisuuksien ja muodostimien perusteella. Malliosion keskeisten luokkien luokkadiagrammit on lisätty selkeyttämään eri luokkien välisiä yhteyksiä ja siitä, kuinka muuttujia on luotu toisen luokan ominaisuuksien pohjalta.



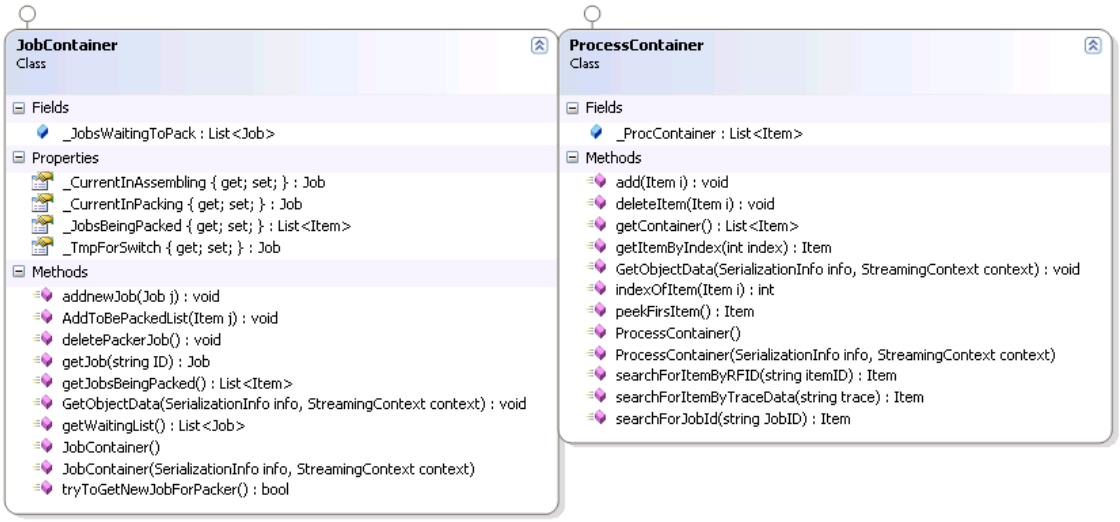
Kuva 7. Tilaustietojen tallennukseen tarkoitettujen luokkien esittely.

Laitteen valmistuksen eri vaiheita ovat mallin näkökulmasta yksinkertaistettuna kokoonpano, prosessissa, pakkaamista odottavat ja pakattavana. Kokoonpano- ja pakkausvaiheissa voi kerrallaan olla vain yksi tuote, ja siksi ne kerätäänkin niille määrättyihin sen hetkisiin muuttujiin. Prosessissa olevia ja pakkaamista odottavia tuotteita voi olla samanaikaisesti useampia, joten ne kerätään listamuotoisiin muuttujiin. Tietokannan palauttamat tilaustiedot ovat talletettuna kuvan 7 OrderContainer-luokassa sijaitsevaan ObservableCollection<OrderQueryResult>-tyyppiseen _OrderContainer muuttujakokoelmaan. OrderQueryResult on Vaisala.VTX–kirjastossa oleva luokka johon tietokannan palauttaman tiedon rakenne perustuu. OrderContainer-luokkaan talletetaan kaikki tietokantahaussa käytetyt parametrit. Haun palauttamista tilaustiedoista kerätään tarpeellisimmat muuttujaan _OrderResultInfo. Se on olio luokasta OrderResultInfo, eli muuttuja, joka on luotu kuvassa 7 olevan OrderResultInfo-luokan pohjalta. OrderContainer-luokka on myös hallittavissa App-luokasta käsin, koska siitäkin on luotu olio ohjelman alustusten yhteydessä. Malliosion luokkien hallinnointi tapahtuu App-luokasta juuri luokista tehtyjen olioiden avulla.



Kuva 8. Luokkadiagrammit luokista Job ja Item. Näitä luokkia käytetään tuote- ja tilauskohtaisesti tietojen tallentamiseen toivotussa muodossa.

Job-luokka on sovelluksen yksi tiedonhallintaan ja tallennukseen käytettävistä luokista. Kuvassa 8 olevassa Job-luokassa on ominaisuuksia, joihin sijoitetaan laskurien arvoja. Käytössä olevia laskureita ovat esimerkiksi ”kuinka monta tuotetta on tilattu”, ”työn alla”, ”pakkauksessa” ja ”valmistettuna”. Kuvan 8 Item-luokan ominaisuuksien pohjalta luotuihin ominaisuuksiin, CurrentAssemblerItem ja CurrentPackerItem, tallennetaan valmistettavien ja pakattavien tuotteiden tietoja. Muuttujia käytetään työn alla olevien tuotteiden tietojen oikeanmuotoiseen tallentamiseen sekä tilaus- ja tuotetietojen linkittymiseen. Item-luokassa on tuotteen lisätietojen tallentamiseen tarpeellisia ominaisuuksia, joiden pohjalta Item-tyyppisiä muuttujia luodaan.



Kuva 9. Tilaukseen kuuluvien tuotteiden tietojen tallennukseen käytettävien luokkien luokkadiagrammit. Kuvan luokkiin tallennetaan sekä hetkellisesti että tilauksen valmistuksen ajan.

Job-luokassa on ominaisuuksia, jotka perivät rakenteensa Item-luokan ominaisuuksien ja muodostimen perusteella. Job-luokassa olevien ominaisuuksien pohjalta luodaan kuvassa 9 olevan JobContainer-luokan Job-muotoisia ominaisuuksia. Niissä säilytetään tiedot pakkaus- ja kokoonpanovaiheissa olevista töistä: CurrentInAssembling ja CurrentInPacking. Näissä muuttujissa on sen hetkisten tuotteiden tietojen lisäksi erilaisia tilauslaskureita. Ominaisuuksien avulla työn alla olevien tuotteiden tiedot saadaan yhdistettyä tilauksen tietoihin. Kyseisen tuotteen vaikutus tuotantotilanteeseen on helpompi ottaa ohjelmassa huomioon, kun muuttujassa on tuotetietojen lisäksi tuotantolaskurien lukemat mukana. Tuotteita poistettaessa prosessista tai asetettaessa niitä luotaukseen on ne helppo poistaa tilauksen tiedoista, kun tuotteen tiedot on linkitetty tilaukseen.

Kun laite on koottu ja laitettu uudelleengenerointiin, siirretään sen tiedot ProcessContainer nimiseen listaan. Kuvassa 9 kuvatussa luokassa on listamuotoinen _ProcContainer-muuttuja, johon on listattuna kokoonpanosta kalibrointivaiheeseen siirtyneet tuotteet. Käytettävä lista on luotu Item-luokan pohjalta. Laitteen tullessa lopputestauksesta pakkauspöydälle siirtyvät myös tuotteen tiedot ProcessContainerista JobsBeingPacked-listaan. Siellä on listattuna tuotteet, jotka ovat pakkausvaiheessa odottamassa lopullista pakkausta. Pahvilaatikkoon pakattua tuotetta pakattaessa tilauslaatikkoon luetaan siihen liimattu viivakoodi. Tilauksen valmistus- ja pakkauslaskureihin päivitetään yhden tuotteen valmistuminen. Tuotteen tiedot jäävät talteen JobsBeingPacked listaan mutta tietoihin merkitään tuote sovelluksen muistissa ”pakatuksi”. Kun

tilauslaatikko täyttyy, eli laatikollisen verran tuotteita on ”pakattu”, annetaan tietokannalle pakkauskomento, jonka jälkeen tuotteet on myös tietokannassa merkitty pakatuiksi. Tämän jälkeen ohjelmassa luodaan uusi laatikko, eli seuravaaksi pakattavat tuotteet määritellään seuraavaan laatikkoon, niitä merkitessä ”pakatuiksi”.

4.3.2 Näkymäosio

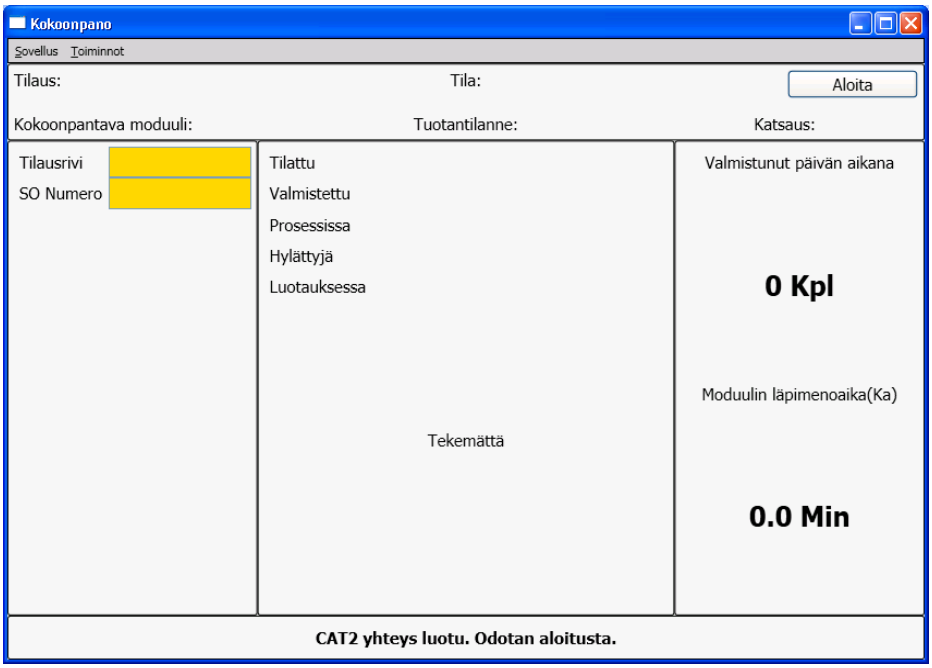
View eli näkymä esittää informaation käyttäjälle. Näkymää hallitsee kontrolleri, joka tulkitsee käyttäjän käskyjä ja niiden mukaisesti hakee malliosiosta tiedot näytettäväksi. Ohjelmassa on kaksi päänäkymää, joiden on tarkoitus toimia kahdella eri näytöllä. Toinen näkymä esitetään kokoonpanovaiheessa ja toinen pakkausvaiheen näytöllä. Näkymissä tulostettavat tekstit ohjeistavat kyseisen vaiheen toiminnoissa ja ilmoittavat tuotannon tilanteesta. Molemmilla näkymillä on ohjelmassa omat luokkansa: PackerWindow ja AssemblerWindow. Nämä luokat perivät MainWindow-luokan, jossa on metodeja näyttöjen tulosteiden päivittämisestä ja infotekstien tulostamisesta värien väläyttämiseen. Näitä metodeja apuna käyttäen molemmat näkymät opastavat tuotannon työntekijää työvaiheissa ja kertovat valmistettavien tuotteiden määrät ja tuotetyypit sekä muita tarpeellisia tietoja. Molempien näkymien omissa luokissa on kyseiselle näkymälle soveltuvia metodeja, joiden avulla näkymien sisällöt saadaan soveltuviksi juuri kokoonpano- tai pakkausvaiheen näkymässä.

Kuva 10. Näkymäosion kahdelle päänäkymälle kuuluvien luokkien luokkadiagrammit, sekä niiden perimä MainWindow-luokka.

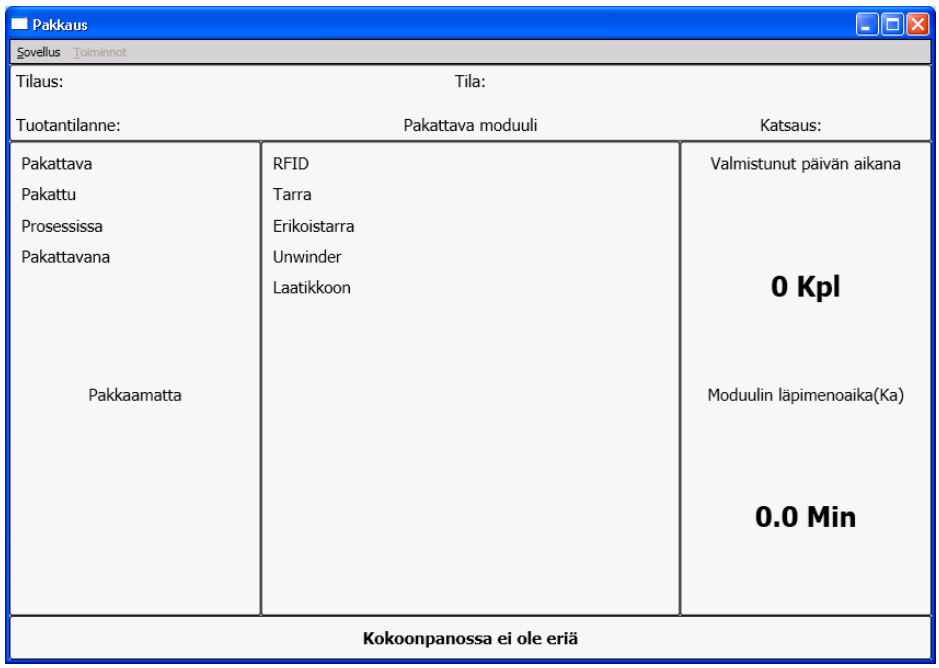
Kuvassa 10 on molemmille päänäkymille kuuluvien luokkien sekä niiden perimän MainWindow-luokan luokkadiagrammit. Kokoonpanonäkymän luokka on nimeltä AssemblerWindow ja pakkausvaiheen näkymän luokka on PackerWindow. Kokoonpano- ja pakkausvaiheiden näkymät on siis luotu siten, että ne perivät MainWindow-luokalta ominaisuuksia ja toimintoja. MainWindow-luokassa on metodeja, joiden avulla

kontrolleri saa hallittua molempia päänäkymiä. Ohjelman App-luokassa on luotuna oliot, joiden avulla näkymä-luokkien metodeja pystyy kutsumaan suoraan App-luokasta. Esimerkiksi ohjeistuksen tulostaminen kokoonpanonäkymässä näytöille on toteutettu kutsumalla kokoonpanonäkymän _AssemblerWindow-olion kautta MainWindow-luokan metodia setAdvisorContent. Koska kokoonpano- ja pakkausnäkymä perivät päänäkymäluokan, voivat ne myös toteuttaa sen sisältämiä metodeja. Molemmilla näkymillä on myös omia toteutuksia, joilla näkymien muokkaaminen onnistuu juuri kyseisen näkymälle tarkoitetuilla toteutuksilla. Esimerkiksi vain kokoonpanovaiheen näkymällä on painike, joten painikkeen tapahtumankäsittelijän toiminnot on toteutettu luonnollisesti vain AssemblerWindow-luokassa.

Vaisalan omassa luokkakirjastossa on olemassa valmis alusta näkymien toteutukseen ja yksi solun ohjaussovelluksen kehitysmahdollisuuksista on toteuttaa näkymät kyseistä mallia pohjana käyttäen. Käyttämällä toteutusmallia pohjana saataisiin myös solunohjaussovelluksen näkymä sopeutumaan muiden tuotannossa toimivien sovellusten ulkoasuun. Tähän sovellusversioon näkymät on toteutettu Windows Presentation Foundation Window-käyttöliittymänä ilman valmiita pohjia. Näkymän objektien hallinta, esimerkiksi näkymien asettelu tai sisällön muokkaaminen, onnistuu ajon aikana vaivattomasti.

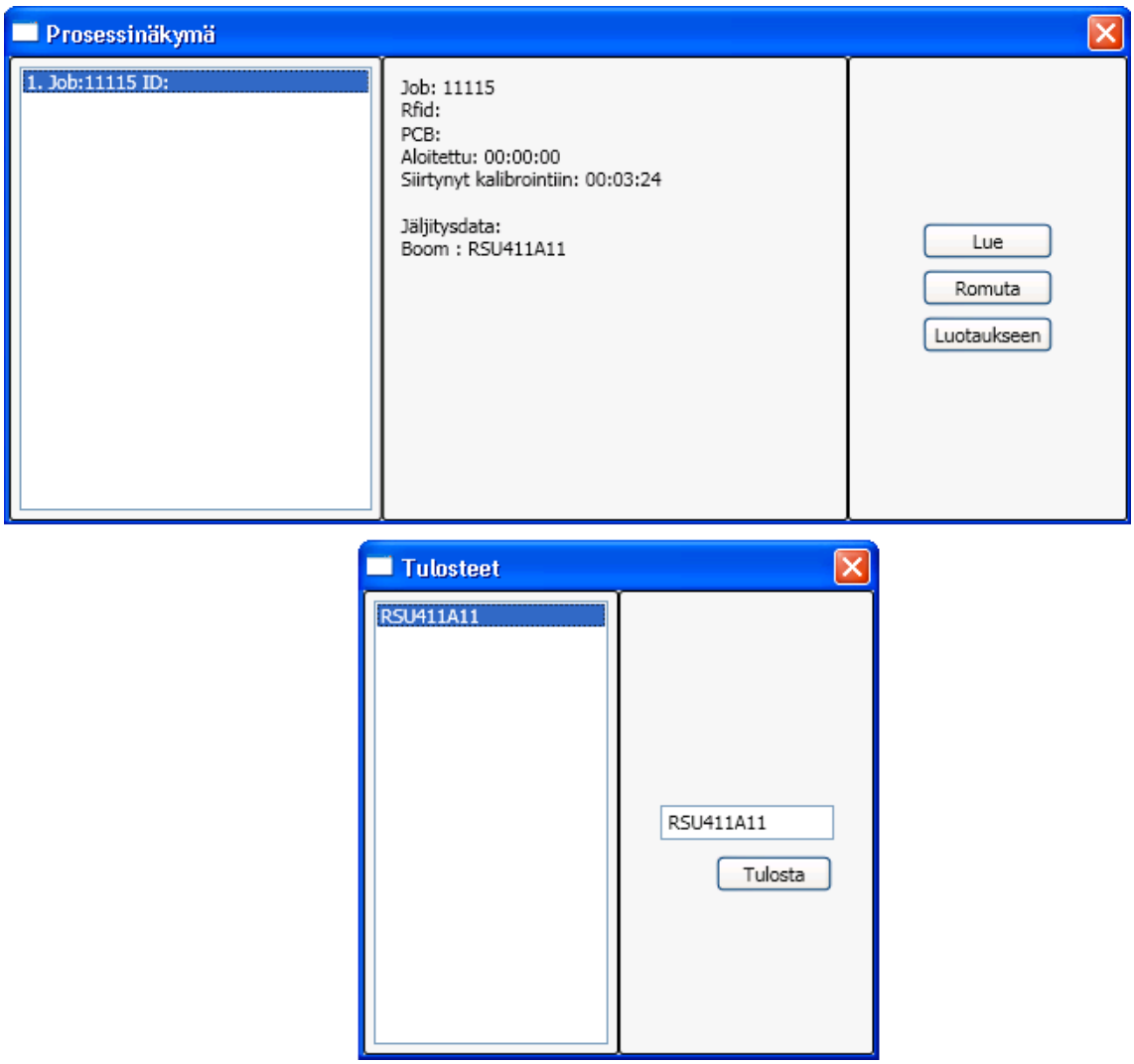


Kuva 11. Kokoonpanovaiheen AssemblerWindow -näkymän alkutilanne



Kuva 12. Pakkausvaiheen PackerWindow -näkymän alkutilanne

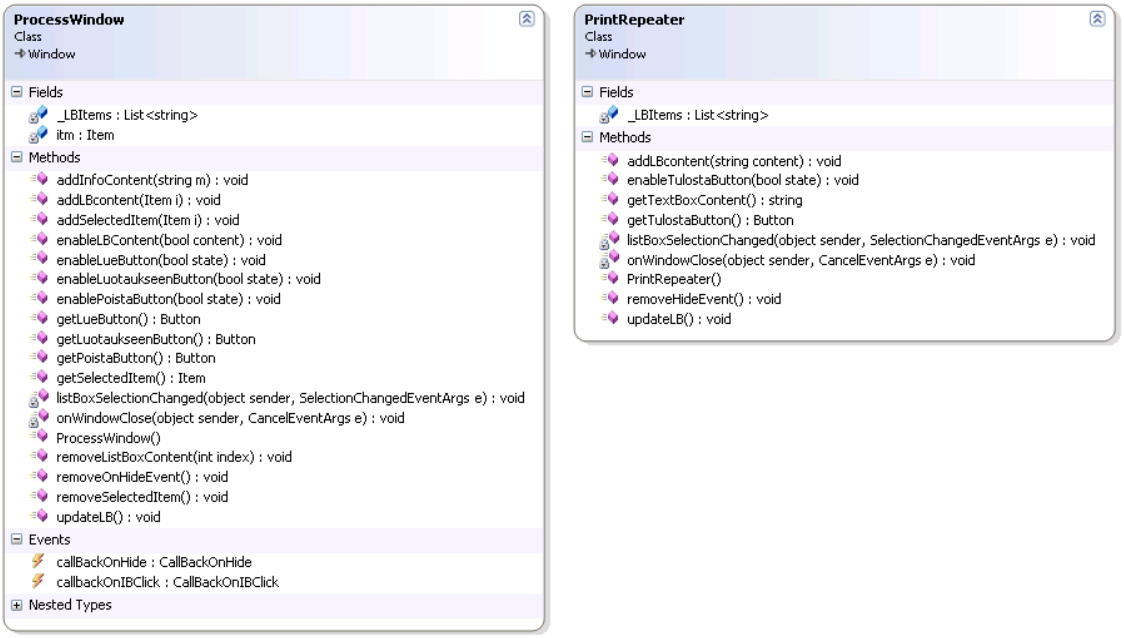
Kuvissa 11 ja 12 on solun ohjaussovelluksen kahden päänäkymän alkutilanne. Nämä ovat näkymiä, jotka käyttäjä saa näytöilleen käynnistäessään sovelluksen. Solun ohjaussovelluksessa molempien näkymien ulkoasu on toteutettu näkymien omien luokkien alustuksissa, joissa molempiin luodaan näkymän ylä- ja alaosaan koko näkymän levyinen paneeli sekä keskiosa jaetaan poikittaissuunnassa kolmeen osaan. Molemmissa näkymissä yläpaneeliin päivitetään ajon aikana tilauksen tietoja ja alapaneeli on tarkoitettu työntekijän ohjeistamiseen joko asettamalla teksti alapaneeliin tai väläyttämällä tekstiä. Oikeassa paneelissa tulostetaan solussa tapahtuneen kyseisen päivän aikana tapahtuneen tuotannon tietoja, kuten päivän aikana valmistuneiden tuotteiden lukumäärä ja keskimääräinen läpivientiaika. Työn alla olevan tuotteen tiedot esitetään kokoonpanon näkymässä vasemmassa ruudukossa, ja pakattavan tuotteen tiedot ovat pakkausnäkymässä keskimmäisessä ruudukossa. Tuotantotilanteen laskurit näytetään kokoonpanossa keskimmäisessä ja pakkauksessa vasemmassa ruudukossa. Tuotantotilanteen laskureita ovat siis tilausmäärä, valmistettujen tuotteiden määrä, prosessissa olevien tuotteiden määrä sekä hylätyt ja luotaukseen ohjatut tuotteet.



Kuva 13. Lisänäkymät Prosessinäkymä ja Tulosteet.

Kahden keskeisen näkymän lisäksi ohjelmassa on kaksi lisänäkymää, jotka saa näkyviin AssemblerView-näkymän työkaluvalikosta. Valikosta Toiminnot löytyy kaksi uutta näkymää: Tulosteet ja Prosessi. Kuvassa 13 esitellyissä näkymissä on tilaukseen lisätty yksi tuote, joka on esimerkkitilanteessa prosessissa eli kiertämässä tuotantolinjaa. Prosessinäkymässä on lista tuotannossa olevista tuotteista, joista tuotteen voi tarpeen mukaan romuttaa tai asettaa sen luotaukseen. Painikkeen kutsuma metodi päivittää tuotantotilanteen laskurit sekä näkymän esittämät lukemat. Lue-toteutus on tarpeellinen jos esimerkiksi romutettavaksi laitettavassa tuotteessa ei ole näkyviä yksilöllistäviä merkintöjä. Tällöin käyttäjä painaa Lue-painiketta ja antaa viivakoodinlukijalla jatkotoimenpiteitä vaativan tuotteen sarjanumeron. Ohjelma hakee prosessista olevista tuotteista kyseisen tuotteen ja asettaa sen romutettavaksi tai luotaukseen. Kuvan 13 Tulosteet-näkymässä on ListBox, johon on lisättynä kaikki tulostetut viivakoodit. Sieltä

ne voi tulostaa valitsemalla listasta tulostettava koodi ja painamalla painiketta. Tulostustoiminto on tarkoitettu ongelmatilanteiden varalta, jotta tulostettuja tarroja saisi tulostettua uudelleen.



Kuva 14. Prosessinäkymän ProcessWindow-luokka sekä Tulosteet näkymän PrintRepeater-luokka luokkadiagrammien avulla esiteltyinä.

Prosessi- ja Tulosteet-lisänäkymien luokat ProcessWindows ja PrintRepeater esitellään luokkadiagrammien avulla kuvassa 14. Luokan metodit, kuten päänäkymien toteutukset, ovat myös App-luokasta kutsuttavissa. Lisänäkymien luokkien metodien avulla saadaan App-luokasta käsin myös lisättyä tietoja näkymien listauksiin tai poistettua tietoja sekä siirrettyä tuotteita pois valmistusprosessista. Ideana on, että vaikka lisänäkymät eivät olisikaan näkyvillä, voi niiden listoihin lisätä tietoja. Jos käyttäjä tarvitsee lisänäkymien ominaisuuksia, ovat tiedot valmiiksi listoissa, eikä niitä esimerkiksi ladata mistään näkymien latautuessa näkyviin.

4.3.3 Käsittelijäosio

Käsittelijäosio käsittää ohjelman toiminnallisuuden ottamalla vastaan ohjelman käyttäjältä saamat käskyt. Kontrolleri muuttaa näkymää ja mallia syötteen mukaisesti. Ohjelma toimii alustusten jälkeen käyttäjän syötteiden perusteella ja näitä syötteitä ovat viivakoodin lukijan syöte, työkalupalkista valitseminen tai painikkeiden painaminen.

Ohjelman tapahtumankäsittelijät on toteutettu delegaateilla eli metodiosoittimilla. Tämä tarkoittaa, että delegaattia kutsuessa kutsutaankin sen osoittamia metodeja. Näitä delegaattia kuuntelevia metodeja voi olla yhtä delegaattia kohden useampia ja näin ollen yksi tapahtuma voi johtaa useamman metodin suorittamiseen. Delegaatin esittely tapahtuu lähes kuin metodin esittely, mutta ilman toteutusta ja delegate-avainsana lisättynä.

```
// Osoittaja ohjaamaan viivakoodinlukijan lukutapahtumankäsittelijän
private delegate void AssemblerfuncPointer(string message);

// Määritellään tapahtumapohjainen delegaatti
private event AssemblerfuncPointer E_assemberInvocation

// Funktio jota kutsutaan
E_assemberInvocation = new AssemblerfuncPointer(startProduction);

// Kutsutaan delegaation osoittamaa metodia
E_assemblerInvocation.Invoke(_AssemblerReader.BarCodeOutput.ToString());
```

Kuva 15. Esimerkkikoodi delegaattien luonnista ja käytöstä.

Kuvassa 15 on esimerkkikoodilla havainnollistettu kuinka delegaattien eri osioita luodaan ja sovelletaan. Delegaattien osoittimia muuttamalla koodin eri vaiheissa saadaan sama tapahtuma kutsumaan eri metodia. Tämä on hyvä ominaisuus esimerkiksi kokoonpanovaiheessa, kun ensimmäisellä kokoonpanodelegaatin kutsulla pyritään aloittamaan tuotanto ja toisella yritetäänkin lisätä tuotteita työn alla olevaan tilaukseen. Koko sovelluksen toiminta perustuu kontrollerin toimintaan ja sen kutsumiin metodeihin. Delegaattien avulla ohjelman suorituksen ohjaus onnistuu selkeäämin, kun ei tarvitse luoda esimerkiksi ehtolauseita toimintojen metodeja kutsuessa. Delegaatin osoitinta muuttamalla saadaan seuraava delegaatin kutsu ohjautumaan oikean metodin suoritukseen.

4.4 Sovelluksen keräämä tieto

Solun ohjaussovelluksen tulee siis kerätä talteen kaikki jäljitysdata, lisätä osat tietokantaan ja linkittää osat lopulliseen tuotteeseen komennolla partlist. Jäljitettäviä tietoja on uudessa radiosondissa tyypillisesti tuotteessa käytettävä kieli, eli liuska, jossa on erilaisia antureita ja elektroniikkalevy. Jäljitettävien tietojen määritykset eivät ole lopullisia, joten niihinkin voi tulla muutoksia. Tämän vuoksi jäljitettävien osien määrittäminen ohjelmalle on oltava konfiguroitavissa. Kokoonpanossa tuote suojataan pehmusteilla,

kuorilla, tiivisteillä sekä siihen liitetään paristot ja unwinder, eli narukerä, jolla sondi on tarkoitus sitoa luotauspalloon. Näistä tuotteista voidaan tulevaisuudessa tarvita jäljitystietoa, joten ohjelman tulee pystyä vastaanottamaan joustavasti seurantatietoa.

Kun tuotantosolun työntekijä käynnistää ohjaussovelluksen, ladataan ensimmäisenä näkymä, jossa pyydetään työntekijää tai työntekijöitä kirjautumaan sisään. Kirjautuminen tapahtuu Vaisalan tunnuksilla. Ohjelma tarkistaa käyttäjätunnus- ja salasanayhdistelmän tietokannasta ja lisää ne sovelluksen niille tarkoitettuun muistipaikkaan. Käyttäjätunnus on tärkeä olla ohjelmassa tallessa, koska tiettyjen tietokantaan annettavien käskyjen yhteydessä metodin parametrina on syötettävä myös toimenpiteen suorittajan tunnus.

Sovelluksen ominaisuuksiin kuuluu myös ajan ottaminen. Ajan ottaminen kuuluu Vaisalan Lean-periaatteeseen, jossa tuotteiden valmistus pyritään saamaan mahdollisimman tuottavaksi. Sovelluksen tuotannon ajastukset ovat tallennettuna tiedostoihin. Tiedostoihin tallennetaan valmistettavan tilauksen tietoja sekä tuotekohtaiset aikaleimat. Jatkokehityksenä voisi olla, että ajat tallennettaisiin tuotantotietokantaan tuotteiden tietojen yhteydessä. Seuraamalla tuotteiden läpivientiaikoja saadaan selkeätä palautetta eri paranteluyritysten vaikutuksista.

Ohjelman ominaisuuksiin kuuluu mahdollisuus keskeyttää tilauksen valmistus sekä taukomahdollisuus. Ohjelman alustusten yhteydessä luodaan tapahtumankäsittelijä, joka ohjaa sovelluksen suorituksen sulkeutumisen yhteydessä sovelluksen metodiin, jossa suoritus lopetetaan hallitusti. Sovelluksen sulkeutuessa tarkistetaan tallessa olevia tietoja ja määritellään, että onko tilauksen valmistus kesken. Jos tietojen perusteella todetaan, että tilaus on kesken, tallennetaan tiedot tiedostoihin. Tiedostot tallennetaan hakemistoon, josta sovellus käy ensimmäisenä tarkistamassa edellisen istunnon mahdollisesti tallentamia tiedostoja. Kokoonpanonäkymässä on lisättynä nappi, jonka avulla tuotanto voidaan laittaa taukotilaan. Taukotilassa yksinkertaisesti ajanottaminen pysäytetään ja odotetaan ohjelman jatkamista.

4.5 Solun ohjaussovelluksen ja CAT2:n välinen liikenne

Yhteys tietokantaan on toteutettu Vaisalan VTX-luokkakirjaston CatServiceProvider luokan avulla. Ohjaussovelluksen vastuulla on tallentaa ajon aikana kerättyjä tuotetietoja

valmistettavista tuotteista sekä tietokantahakujen avulla noutaa tilaustietoja. Työntekijä lukee tilaustiedot tulosteesta, ja solun ohjaussovellus aloittaa ajan ottamisen. Tilauksen muista tietokannasta löytyvistä tilauksista erottavat tiedot ovat tilausnumero ja tilausrivi. Samalla tilausrivillä on tyypillisesti useampia tilausrivejä. Tietokantaan annettavan CatServiceProvider-luokan GetOrders-metodi palauttaa sovellukselle kaikki tiedot, jotka kyseisellä tilausnumerolla löytyy tietokannasta. Metodille on mahdollista antaa parametreina tilauksen muista erottavista tiedoista vain tilausnumero, joten ohjaussovelluksessa on toteutettu hakutoiminto, joka valikoi oikean tilausrivin kaikista tilausnumerolla löytyneistä tilausriveistä. Aiemmin tilauksen käsittelyä aloittaessa on Lean-tulosteessa ollut vain tilausnumero, mutta siihen joudutaan nyt lisäämään myös tilausrivi. Kun oikeat tilaustiedot on löydetty, haetaan valmistettavan tuotetyypin perusteella asetustiedostoista laitekohtaiset asetukset ja sovellus käynnistää tuotannon.

Ajonaikaisesti kerättyä tuotetietoa tallennetaan, kun yksittäisen tuotteen kaikki jäljitettävät tiedot on otettu talteen. Lopullisen tuotteen tuotetyyppi saadaan tietokannan saaduista tilauksen tiedoista ja tässä versiossa solun ohjaussovelluksesta tuotteen sarjanumero määräytyy tuotteeseen liitettävän anturikielen sarjanumerosta. Tuotteilla on mahdollista olla identtiset sarjanumerot, kunhan tuotetyyppi- ja sarjanumeroyhdistelmät ovat yksilöllisiä. Lopullinen tuote tallennetaan tietokantaan VTX-kirjaston CatServiceProvider-luokan UutInfo-tyyppisellä muuttujalla. Tähän muuttujaan lisätään laitteen tuotetyyppi ja sarjanumero. Ennen tallentamista tietokantaan tuotteeseen linkitetään jäljitettävien osien tiedot UutInfon metodilla AddChild. Lopulta tuotetiedot tallennetaan kantaan CatServiceProvider-luokan tarjoamalla toteutuksella.

Kun yksi laatikollinen tai koko tilausmäärä on saatu kokoonpantua, kalibroitua ja pakattua, antaa solun ohjaussovellus kantaan pakkausilmoituksen. CatServiceProvider-luokan Pack-metodilla annetaan tietokantaan ilmoitus, että kyseiset tuotteet on pakattu. Metodin parametreina käytetään tarpeellisia tietoja, jotta juuri kyseinen tilauslaatikko erotetaan muista laatikoista, ja laatikon sisältö on tarkasti tiedossa tietokannassa. Pakkauslaatikosta muistissa olevia tietoja ovat tilausnumero, tilausrivi, pakkauslaatikon numero, tuotteiden tuotetyyppi, pakkaajan nimikirjaimet, pakkauspäiväys ja lista pakattujen tuotteiden sarjanumeroista.

5 Jatkokehitysmahdollisuudet

Ohjelman kehitysvaiheessa vaatimukset ja toivotut ominaisuudet lisääntyivät jatkuvasti, joten insinöörityö päädyttiin rajaamaan kehitysvaiheessa olevaan sovellusversioon. Ilmaantuneet kehitysmahdollisuudet ovat varmasti sellaisia, joita lähdetään sovellukseen toteuttamaan heti demotilaisuuden jälkeen. Useimmat jatkokehitysmahdollisuudet tulivat kaizen-suunnittelutapahtumissa ja joitakin saatiin ihan kuulemalla tuotannon työntekijöiden mielipiteitä. Lisää ideoita tulee varmasti lisää, mitä lähemmäs päästään sovelluksen käyttöönottoa ja sen jälkeenkin.

Suurin kehitysmahdollisuus on ohjaussovelluksen joustavuudessa. Sovellusta on alusta asti kehitetty siten, että sitä käyttämällä olisi mahdollista valmistaa hyvin erilaisia tuotteita. Eri tuotteiden valmistuksen yhteydessä ongelmiksi muodostuu juuri tuotekohtaisten jäljitettävyystietojen vaihtelevuus. Sovelluksen joustavuutta lisäävä ominaisuus, on mahdollisuus lisätä lopputuotteeseen jäljitettäviä osia pakkausvaiheessa. Tässä sovelluksessa jäljitettävät osat on luettava ja tallennettava tietokantaan osalistaukseen ennen kuin laitteen voi siirtää seuraavaan vaiheeseen. Kaivattu ominaisuus lisäisi mahdollisuuden lisätä vielä pakkauspöydällä jäljitettäviä osia tuotteeseen. Pakkausvaiheessa tuotteeseen lisättävä osa voi olla esimerkiksi unwinder, eli narukerä, jonka toinen pää on luotauspallossa ja toinen sondissa. Tällä hetkellä ohjaussovelluksessa ei ole mahdollista lisätä jäljitettäviä osia pakkausvaiheessa.

Kun tilauslaatikko on saatu täyteen, olisi tarkoitus luoda Vaisalan luokkakirjaston avulla raportti, joka sisältäisi sarjanumerolistauksen sekä mahdollisesti muita tietoja tilauksesta. Tämän tulosteen voi sitten laittaa mukaan liitteenä tilauksen pakkauslaatikkoon tai tallentaa PDF-tiedostoksi.

Tuotantosoluun on myöhemmin tulossa lopullinen sondien tuotannossa toimivat tarratulostin. Tulostinta lisättäessä sovellukseen tullaan tulostustoiminnot toteuttaa uusiksi. Uuden tulostimen yhteydessä tulee luoda tulostimelle laiteajuri, jolla onnistuu tulostimen käyttäminen ohjelmistokoodissa. On mahdollista, että tuleva tarratulostin on jo ennestään yrityksessä käytössä olevaa mallia. Tällöin ei tarvitse kirjoittaa laiteajureita täysin uudestaan, vaan voidaan soveltaa yrityksen sisäisen luokkakirjaston tarjoamaa ajurikirjastoa.

Toiminnallisia muutoksia on työn alla olevan tilauksen keskeyttäminen ja keskeytetyn ohjelman uudelleenkäynnistäminen. Tässä on ideana, että jos solussa on työn alla tilaus, mutta kiireellisempi tilaus vaatii toteuttamista, on uuden työn aloittaminen oltava mahdollista. Tämänhetkinen toteutus osaa ottaa talteen suljettavan istunnon tuotantotilanteen ja käynnistää myöhemmin saman istunnon. Meneillään olevan tilauksen sulkeminen toisen tilauksen alta ei ole tällä hetkellä mahdollista. Sovelluksella ei ole myöskään mahdollista aloittaa uuden tilauksen käsittelyä ennen kuin edellinen tilaus on saatu täysin valmistettua. Toivottu ominaisuus mahdollistaisi tuotannossa jatkuvan täyden virtauksen, eli tilauksen vaihtuessa toiseen ei tulisi turhaan tyhjiä työvaiheita.

Yksi kehitysmahdollisuus on solun ohjaussovelluksen kyky keskustella tai vähintään virheidenhallinta rinnakkaisissa soluissa toimivien samankaltaisten solun ohjaussovellusten välillä. Samanlaisia tuotantosoluja on tulossa useampia tuotantoalueelle, ja solun ohjaussovellusten tulee pystyä toimimaan ongelmattomasti rinnakkain. Ongelmia voi aiheuttaa esimerkiksi soluissa työn alle otettavat työt, jotka voivat olla samoja useammassa solussa. Ohjaussovelluksen virheenhallinnan tulee toimia, jos joitakin ongelmatilanteita ilmenee useamman ohjaussovelluksen käytössä.

Ohjaussovelluksen näkymien toteutusta on mahdollista kehittää Vaisalan mallipohjan avulla selkeämmäksi. Vaisalan VTX-luokkakirjastossa on osio, jossa on valmiita metodeja ja toimintoja näkymien hallintaan ja muokkaamisiin. VTX.Main on valmis mallipohja käyttöliittymän toteutukseen, jonka pohjalta toteutuksen käyttöliittymien toteutus olisi mahdollista.

Ohjelman toimintaan liittyviä asioita mietittiin vielä kehitysvaiheessa. Alun perin tuotantoa aloitettaessa luettiin viivakoodinlukijoilla vain tilausnumeron viivakoodi. Tämä toteutus ei kuitenkaan ollut riittävä, koska radiosondien tilauksissa yhtä tilausnumeroa kohden voi olla useampia tilausrivejä, joista tuli pystyä tunnistamaan oikea. Tämä johti tuotannossa käytössä olevan Lean-paperin tulosteen uusimiseen. Tulosteen ulkoasun muuttumisen vuoksi pohdittiin koko tulosteen tuonnin uusimista. Uudessa tavassa tulostamisen sijaan tilauksen tiedot kopioitaisiin suoraan XML-tiedostoon, josta solu saisi kaiken tarvittavan tiedon tilauksen aloittamiseen. Tilaustietojen kirjoittaminen tiedostoon lisäisi tuotannon tehokkuutta papereiden siirtelyn vähetessä. Tämä tapa osoittautui kuitenkin liian suureksi työksi, koska kyseinen toteutus vaikuttaisi liikaa koko

tuotannon toimintaan. Tilauksen tuontia tuotantosoluun aloitettavaksi tullaan jatkossa suunnittelemaan myös erillään tästä solun uudistusprojektista.

6 Päätelmät

Työn tavoitteena oli luoda kehityskelpoinen ohjaussovellus Vaisalan uudistuvaan tuotantosoluun. Sovelluksen kehitysvaiheessa vaatimuksien ja ominaisuuksien määritelmät laajenivat sen verran, että työ jouduttiin rajaamaan sovelluksen kehitysvaiheessa olevaan versioon. Sovelluksen kehityksen edetessä ohjelmaan tarvittuja ominaisuuksia tuli jatkuvasti lisää, joten niiden toteuttaminen päätettiin ajoittaa myöhemmälle ajankohdalle. Kehitysvaiheessa oleva sovellus on riittävä tässä vaiheessa projektia, koska tuotantosolukin on vielä työn alla. Jatkokehitysmahdollisuuksia tullaan toteuttamaan ohjaussovellukseen samalla kun solun suunnittelu etenee.

Vaikka kehitettävää jäi runsaasti, on sovellus kuitenkin alustavien vaatimusten mukainen. Sovellukselta onnistuu kaikki tarvittava tietokantaliikenne ja tiedonkeruu, jotta tuotanto voitaisiin käynnistää. Tuotannon ohjaus on toteutettu ohjaussovelluksessa, mutta ohjauksessa käytettävät ilmoitukset ja komennot tulevat tarkentumaan projektin edetessä. Aikaansaadussa versiossa ovat ohjeistukset alkuperäisen vaatimusten mukaisesti. Jatkokehityskin on Lean-ajattelumallin mukaisesti koko ajan jatkuvaa ja varmasti näiden kehitysmahdollisuuksien toteuduttua ilmenee lisää parannettavia kohtia. Kuten tuotantosolu ei solun ohjaussovelluskaan välttämättä ole koskaan lopullisesti valmis.

Lähteet

1 Lean Manufacturing Articles and Resources. 2009. Verkkodokumentti. Gembutsu. <http://www.gembutsu.com/lean_resources.html>. Luettu 26.4.2012.

2 .NET Framework 4. 2012. Verkkodokumentti. Microsoft. <http://msdn.microsoft.com/en-us/library/w0x726c2.aspx>. Luettu 1.5.2012.